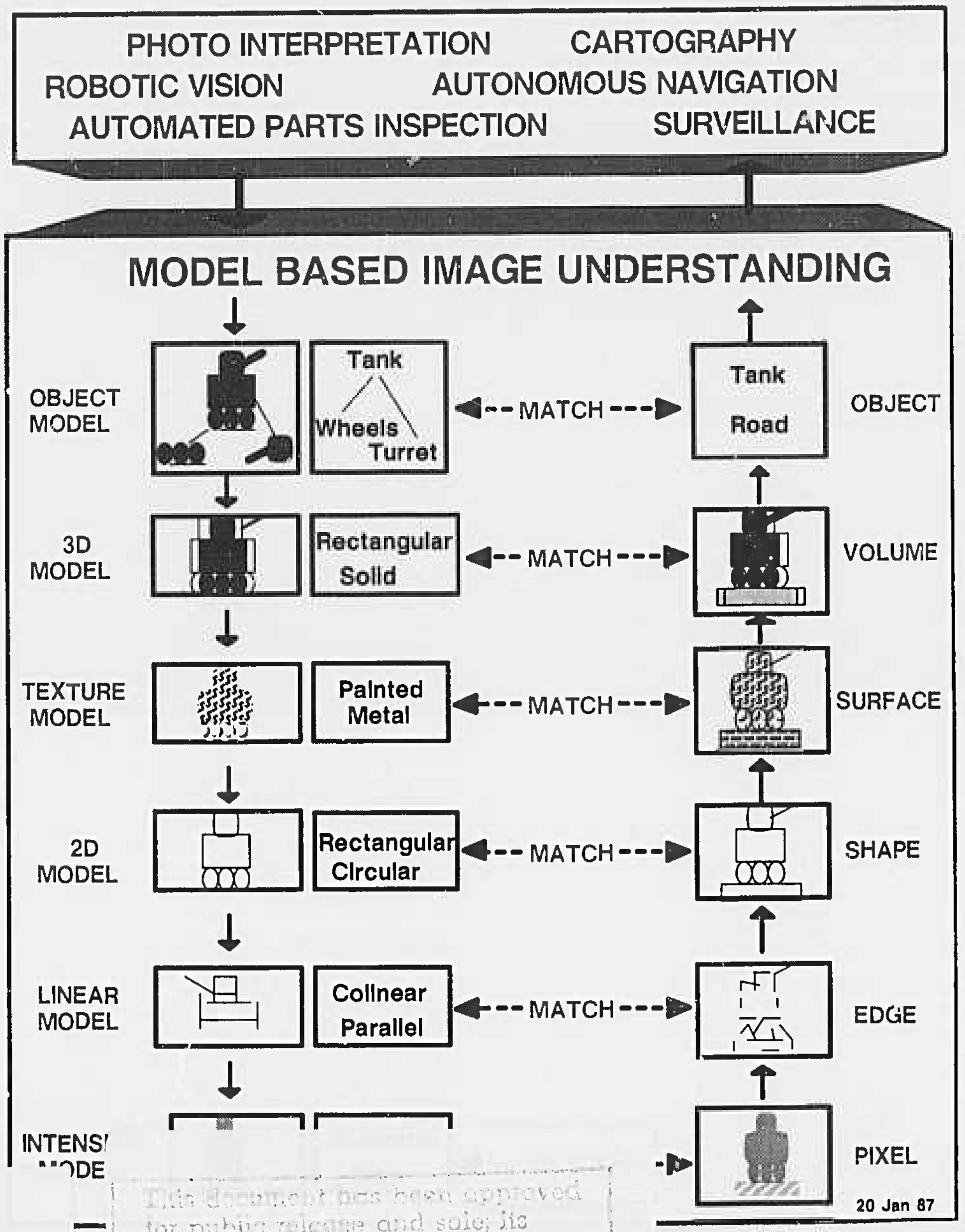
(2)

Volume I

PROCEEDINGS: Image Understanding Workshop

AD-A186 103



DTIC ELECTE JUL 2 2 1987 S D A

This document has been approved for public release and sale; its distribution is unlimited.

87 7 21 152

Sponsored by:
Defense Advanced Research Projects Agency
Information Science and Technology Office

DARPA

SC

February 1987

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A186103	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Proceedings: Image Understanding Workshop February 1987		5. TYPE OF REPORT & PERIOD COVERED ANNUAL TECHNICAL December 1985-February 87
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Lee S. Baumann (Editor)		8. CONTRACT OR GRANT NUMBER(s) N00014-86-C-0700
9. PERFORMING ORGANIZATION NAME AND ADDRESS SCIENCE APPLICATIONS INTERNATIONAL CORP 1710 Goodridge Drive McLean, VA 22102		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS ARPA Order 5605
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209		12. REPORT DATE February 1987
		13. NUMBER OF PAGES 1000
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE, DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Digital Image Processing; Image Understanding; Scene Analysis; Edge Detection; Image Segmentation; CCDArrays; CCD Processors

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This document contains the outlines of annual progress reports and technical papers presented by the research activities in Image Understanding, sponsored by the Information Science & Technology Office; Defense Advanced Research Projects Agency. The papers were presented at a workshop conducted on 23-25 February 1987, in Los Angeles, California. Also included are copies of invited papers presented at the workshop and additional technical papers from the research activities which were not presented due to lack of time but are germane to this research field.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Keywords includ

Image Understanding Workshop

Proceedings of a Workshop
Held at
Los Angeles, California

February 23-25, 1987

Volume I

Sponsored by:

Defense Advanced Research Projects Agency
Information Science and Technology Office

This document contains copies of reports prepared for the DARPA Image Understanding Workshop. Included are results from both the basic and strategic computing programs within DARPA/ISTO sponsored projects.

APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied of the Defense Advanced Research Projects Agency or the United States Government.

TABLE OF CONTENTS

QUALITY INSPECTED 4

Accession For
NTIS GRA&I ✓
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

Page

TABLE OF CONTENTS

TABLE OF CONTENTS

<u>AUTHOR INDEX</u>

AUTHOR INDEX (Continued)

AUTHOR INDEX (Continued)

AUTHOR INDEX (Continued)

<u>AUTHOR INDEX</u> (Continued)

SECTION I

PROGRAM REVIEWS BY

PRINCIPAL INVESTIGATORS

VISION IN DYNAMIC ENVIRONMENTS

Azriel Rosenfeld
Larry S. Davis
John (Yiannis) Aloimonos

Center for Automation Research
University of Maryland
College Park, MD 20742

ABSTRACT

This report briefly summarizes research carried out during 1986 under Contract DAAB07-86-K-F073, monitored by the U.S. Army Center for Night Vision and Electro-Optics (COTR: Dr. George Jones).

1. INTRODUCTION

The Computer Vision Laboratory of the Center for Automation Research at the University of Maryland has been funded under the DARPA Image Understanding Program since 1976. The current contract, entitled "Vision in Dynamic Environments", was initiated in June 1986. This report briefly summarizes research carried out under this contract, as well as work completed since the end of 1985 that was initiated under a predecessor contract.

The research described here is concentrated in three main areas: motion analysis; stereo and range sensing; and three-dimensional shape. A bibliography of technical reports issued on the contract (and its predecessor) during the period November 1985 – December 1986 is appended to this report.

The descriptions given here are quite brief, but several of the reports are included in these Proceedings as technical papers. In addition, two papers describing recent work, dealing respectively with active vision and with learning shape computations, are included in these Proceedings. Three other papers describing other work done at the University of Maryland are also included; they deal, respectively, with autonomous land vehicle navigation, road network detection in aerial images, and computational techniques for rapid recognition of familiar, but unexpected, objects.

2. MOTION ANALYSIS

In the area of motion analysis, extensive work was done during 1985–6 at the University of Maryland by Prof. Ken-ichi Kanatani of Gunma University, Japan. This work dealt with the following topics:

a) <u>Analysis of Structure and Motion from Optical Flow, Part I: Orthographic Projection</u>

The 3D structure and motion of an object is determined from its optical flow under orthographic projection. First, the image domain is divided into planar or almost planar regions by checking the flow. For each region, parameters of the flow are determined. Transformation rules under coordinate changes and hydrodynamic analogies are also discussed. The 3D structure and motion are determined in explicit forms in terms of irreducible parameters deduced from group representation theory. The solution is not unique, containing an indeterminate scale factor and comprising true and spurious solutions. Their geometrical interpretations are also studied. The spurious solution disappears if two or more regions of the object are observed.

b) <u>Analysis of Structure and Motion from Optical Flow, Part II: Central Projection</u>

In this Part 2, the 3D structure and motion of an object is determined from its optical flow in central projection. As in Part 1, the image domain is divided into planar or almost planar regions by checking the flow. For each region, parameters of the flow are determined. In our flow–based approach, the 3D structure and motion are computed from the irreducible parameters, which are complex numbers in general, deduced from group representation theory. The transition from central projection via "pseudo-orthographic projection" to orthographic projection is also discussed. The solution is not unique. Besides the absolute depth being indeterminate, there arise two solutions, the true one and a spurious one. However, the spurious solution disappears if two regions of the object are observed. The adjacency condition of two planar regions is also studied in terms of complex variables. The relation to the correspondence–based approach is shown, too.

c) <u>Transformation of Optical Flow by Camera Rotation</u>

The effect of camera rotation on the description of optical flow is analyzed. The transformation law of the parameters is explicitly given by considering infinitesimal generators and irreducible reduction of the induced representation of the 3D rotation group. The parameter space is decomposed into invariant subspaces, and the optical flow is accordingly decomposed into two parts, from which an invariant basis is deduced. A procedure is presented to test the equivalence of two optical flows and to reconstruct the necessary amount of camera rotation. The relationship with the analytical expressions for 3D recovery is also discussed.

d) Structure from Motion without Correspondence: General Principle

A general principle is given for detecting 3D structure and motion from an image sequence without using point-to-point correspondence. The procedure consists of two stages: (i) determination of the *flow parameters*, which completely characterize the motion of the planar part of the object, and (ii) computation of 3D recovery from these flow parameters. The first stage is done by measuring *features* of the image sequence. The second stage is analytically expressed in terms of invariants with respect to coordinate changes. Typical features and relations to stepwise tracing are also discussed.

e) Tracing Finite Motions Without Correspondence

The 3D motion of an object having planar faces is traced, starting from a known position, from a sequence of 2D perspective projection images *without using any knowledge of point-to-point correspondence*. Computation is based on the shape of the region on the image plane corresponding to a planar face of the object. Given two images, a heuristic guess about the motion is first computed, and one image is transformed according to this estimated motion so that it is positioned close to the other image. Then, the motion that accounts for the remaining small discrepancy is estimated by measuring numerical *features* of the planar regions. The scheme is based on the optical flow due to infinitesimal motion, and estimation is done by solving a set of simultaneous *linear* equations. This process is iterated; after each estimate of the motion, one image is transformed according to the estimated motion so that it is positioned closer and closer to the target image. Various practical issues such as choice of features, constrained motions, face identification, and computation of features without actually transforming the images are discussed. Some numerical examples are also given.
This paper appears in full in these Proceedings.

Another major effort in motion analysis, under the direction of Prof. Allen M. Waxman (now at Boston University), led to the Ph.D. dissertation of Dr. Muralidhara Subbarao. The following were the specific areas investigated:

f) Interpretation of Image Motion Fields: A Spatio-Temporal Approach

In this paper we describe a new formulation of the image motion interpretation problem. The formulation addresses the general problem of recovering the 3D local surface structure, motion and deformation of an opaque object. It is based on the assumption of local analyticity of 3D surface structure, motion, deformation and, consequently, the corresponding 2D image motion in the space-time domain. In this approach both spatial and temporal information are used in a uniform way. The formulation is very general in the sense that as long as the analyticity assumption is valid, the space and time dependence of surface structure and motion can be related to the image motion parameters. We illustrate our approach by formulating and solving the image motion interpretation problem for some simple cases including non-rigid and non-uniform motions. However, it can be easily extended to deal with more complicated cases.

For rigid and uniform motions we have solved the problem for three important cases. The first two relate to the case where the image motion is observed in a fixed image neighborhood and the other case is where the camera tracks a fixed point on the object in motion and the tracking motion of the camera is known. In all these three cases we have solved for the local orientation and rigid motion of the surface patch around the line of sight using only the first-order spatial and temporal derivatives of the image velocity field. In comparison, all the existing methods based on image motion fields use up to second-order spatial derivatives of the image velocity field which are relatively sensitive to noise.

g) Closed-Form Solutions to Image Flow Equations for 3-D Structure and Motion. (Allen M. Waxman, Behrooz Kamgar-Parsi, and Muralidhara Subbarao)

A major source of three-dimensional (3-D) information about objects in the world is available to the observer in the form of time-varying imagery. Relative motion between textured objects and observer generates a time-varying optic array at the image, from which image motion of contours, edge fragments and feature points can be extracted. These dynamic features serve to sample the underlying "image flow" field. New, closed-form solutions are given for the structure and motion of planar and curved surface patches from monocular image flow and its derivatives through second order. Both planar and curved surface solutions require at most the solution of a cubic equation. The analytic solution for curved surface patches combines the transformation of Longuet-Higgins and Prazdny with the planar surface solution of Subbarao and Waxman. New insights regarding uniqueness of solutions also emerge. Thus, the "structure-motion coincidence" of Waxman and Ullman is interpreted as the "duality of tangent plane solutions." The multiplicity of transformation angles (up to three) is related to the sign of the Gaussian curvature of the surface patch. Ovoid patches (i.e., bowls) are shown to possess a unique transform angle, though they are subject to the local structure-motion coincidence. Thus, ovoid patches almost always yield a unique 3-D interpretation. In general, ambiguous solutions can be resolved by requiring continuity of the solution over time.

h) Interpretation of Image Motion Fields: Rigid Curved Surfaces in Motion

This paper is concerned with recovering the three-dimensional shape and motion of a rigid surface from its image motion field on a camera's image plane. A partial solution to this problem was proposed by Longuet-Higgins and Prazdny, and recently a more complete solution has been obtained by Waxman, Kamgar-Parsi and Subbarao. Here we reconsider this problem in the context of our recent work where a general formulation and solution procedure is proposed for the interpretation of image motion fields. Using this new approach, closed-form solutions are derived for the motion, orientation and curvatures of a rigid surface. In comparison with the previous approaches this approach does not involve rotating the image coordinate system in order to solve for the unknowns. The solution is obtained directly in the original coordinate system, thus saving some computation. More importantly we state and prove some fundamental theoretical results concerning the existence of multiple interpretations for an instantaneous image motion field resulting from a rigid body motion. *Conditions for the occurrence of up to four (four being the maximum possible) solutions are stated and proved.* Numerical examples are given for some interesting cases where multiple solutions exist. The results are presented in a sequential order which suggests a straightforward implementation of the solution method. This work, along with our previous work, suggests a unified computational approach for the interpretation of image motion fields in a variety of situations (e.g.: planar/curved surfaces using spatial and/or temporal image flow parameters, rigid/non-rigid motion, etc.).

i) Interpretation of Visual Motion: A Computational Study

A changing scene produces a changing image or *visual motion* on the eye's retina. The human visual system is able to recover useful three-dimensional information about the scene from this two-dimensional visual motion. This report is a study of this phenomenon from an information processing point of view. A *computational theory* is formulated for recovering the scene from visual motion. This formulation deals with determining the local geometry and the rigid body motion of surfaces from spatio-temporal parameters of visual motion. In particular, we provide solutions to the problem of determining the shape and rigid motion of planar and curved surfaces and characterize the conditions under which these solutions are unique. The formulation is generalized to the case of non-uniform (i.e. accelerated) and non-rigid motion of surfaces. This serves to address the two fundamental questions: *What scene information is contained in the visual motion field? How can it be recovered from visual motion?* The theory exposes the well known fact that the general problem of visual motion interpretation is inherently ill-posed. Furthermore, it indicates the minimum number of additional constraints (in the form of assumptions about the scene) necessary to interpret visual motion. It is found that, in general, the assumption that objects in the scene are rigid is sufficient to recover the scene uniquely.

A *computational approach* is given for the interpretation of visual motion. An important characteristic of this approach is a uniform representation scheme and a unified algorithm which is both flexible and extensible. This approach is implemented on a computer system and demonstrated on a variety of cases. It provides a basis for further investigations into both understanding human vision, and building machine vision systems.

3. STEREO AND RANGE SENSING

The following projects dealt with stereo, range sensing, and related topics.

a) Subpixel Registration (Eliahu Wasserstrom)

A method of subpixel image registration is proposed that employs a model of the correlation surface in the vicinity of the registration point.

b) Experiments in Stereo Matching using Multiresolution Local Support (Fang-Jyh Lin Liu, Roger Eastman and Larry S. Davis)

This paper describes a set of computational algorithms for stereo matching based on multiresolution local support. The algorithms combine the feature-point based coarse-to-fine matching of Marr-Poggio-Grimson, the local support measures of Pollard-Mayhew-Frisby and Prazdny, and other disambiguation techniques. The matching primitives are zero-crossing points. Matching compatibility is based on the sign of the contrast and the gradient direction at the Laplacian zero-crossing point. These algorithms use coarse resolution disparity to constrain the disparity search window in the fine resolution matching process, which speed up the search and greatly improve the accuracy of matching. The consistency measure includes Gaussian local support or disparity gradient threshold local support in combination with symmetric or iterative disambiguation techniques. This paper includes experiments performed on a variety of stereo images containing synthetic, laboratory and aerial scenes.

c) Structured Light Patterns for Robot Mobility (Jacqueline Le Moigne and Allen M. Waxman)

In order to assess the feasibility of using a structured-light range sensor for mobile outdoor and indoor robots, we discuss a number of operational considerations and image processing tools relevant to this task domain. In particular, we address the issues of operating in ambient lighting, smoothing of range texture, grid pattern selection, albedo normalization, grid extraction and coarse registration of image to projected grid. Once a range map of the immediate environment is obtained, short range path planning can be attempted.

d) Road Following by an Autonomous Vehicle Using Range Data (Uma Kant Sharma and Larry S. Davis)

This paper describes a road following system for an Autonomous Land Vehicle. Range data is used as the sensor input. The system is divided into two parts: low-level data driven analysis followed by high-level model-directed search. The sequenee of steps performed, in order to detect 3-D road boundaries, is as follows: Range data is first eonverted from spherieal into Cartesian coordinates. A quadric (or planar) surface is then fitted to the neighborhood of each range pixel, using a least square fit method. Based on this fit, minimum and maximum principal surface curvatures are computed at each point to detect edges. Next, using Hough transform techniques 3-D local line segments are extracted. Finally, model directed reasoning is applied to detect the road boundaries.

e) Practical Computation of Pan and Tilt Angles (Behrooz Kamgar-Parsi)

The sensitivity of the 3-D recovery from a stereo pair of images of object points in space to the errors in the pan and tilt angles is studied. It is shown that precise knowledge of the relative pan angle of the two cameras with respect to each other (and to a lesser degree the relative tilt angle) is crucial to the accurate 3-D recovery of object points in space, whereas accurate knowledge of the pan and tilt angles relative to the scene, i.e. the "world" angles, is of less significance. This indicates that the most important task would be the computation of the relative pan and tilt angles. Theoretically, it is well known that using corresponding left and right image positions, one can calculate the orientation of the two cameras. Limited resolution capabilities of existing cameras, however, makes this task a difficult one. Practical difficulties in computation of camera orientations are discussed. It is shown that while computation of the relative tilt angle is fairly easy, computation of the relative pan and in particular the world pan angles are difficult. Indeed for many image pairs it may not be possible to compute the world pan angle with any degree of reliability. However, it is shown that often it is possible to bypass the computation of the world pan angle and to compute the relative pan and tilt angles directly. This is despite the fact that in the analytic formulation of the problem the three angles are coupled. The stereo model studied here is assumed to have a fixed baseline and small relative pan and tilt angles. A possible application of such a stereo model is the visual system of an autonomous vehicle whose task is road following.

A revised version of this paper appears in these Proceedings.

f) Monocular Stereopsis: Theory and Applications (John Aloimonos and Azriel Rosenfeld)

A theory of monocular depth perception is presented. A moving cyclopean observer uses motion information to recover the depth of an object. The problem is studied for both orthographic and perspective projection and closed form solutions for the absolute depth functions are developed. Finally, an application of the theory to a vibrating camera is presented. In particular, our results are:

(1) A moving cyclopean observer that does not know his motion can recover the absolute depth of an object from its shape and the induced optic flow field. Under orthography, a closed form solution for the absolute depth function is given.

(2) A moving cyclopean observer that knows his motion can recover the absolute depths of objects using only the spatiotemporal derivatives of the image intensity function. This result gives rise to useful applications, for example, the recovery of depth from a vibrating camera.

g) An Efficient Line Search Algorithm for Optimization of Multivariate Functions (Behrooz Kamgar-Parsi)

An efficient line search algorithm for the minimization of multivariate functions is presented. The main element of this algorithm is a new interpolation technique. This interpolation technique is a cubic interpolation which, in addition to the three pieces of information used for a quadratic interpolation, employs the function value in the middle of the interval of interest. In a program based on a quasi-Newton method with the BFGS update formula, we tested the new line search routine against a line search routine that utilizes standard quadratic and cubic interpolation techniques. We used ten standard small-to-medium-size test functions ($2 \leq N \leq 30$). The results indicates that, on the average, when using the new line search routine, the minimization of a function achieved during every two iterations *that require a line search* is equivalent to the minimization of the function which is achieved during three iterations when using a line search routine that is based on standard interpolation techniques. (This suggests that the new line search routine is 33% more efficient.) As regards the general impact of the new interpolation technique on the program, we found the following; reduction in the number of iterations by almost 10%; reduction in the number of gradient evaluations by over 8%; and although the new interpolation technique requires an extra function evaluation, the number of function evaluations, too, showed a decline (albeit small: 2–3%).

h) Calibration of a Stereo System With Small Relative Angles (Behrooz Kamgar-Parsi and Roger Eastman)

Practical difficulties in the calibration of a two camera stereo system in an uncontrolled environment are studied for the case where the relative orientation angles are small and the distance between the two cameras is known. This is done by deriving explicit analytical solutions for the relative pan, tilt and roll angles in terms of the world pan angle (often referred to as gaze angle) and the coordinates of the image points used in their computation. These solutions allow us a better understanding of the intricacies of the problem of calibration in general. The purpose of this work has been twofold, both practical and theoretical. Its practical purpose is to provide us with a reliable method for the computation of camera orientations when the relative rotation angles are small. Its theoretical purpose is to provide us with insight as to how errors due to quantization and uncertainty in the image center location can affect the computation of rotation angles, so that we can look for ways to minimize their impact. (These findings are likely to be of use even when the relative rotation angles are not small.) In particular it is shown that the sensitivity of the computation of the relative pan and roll angles to the above sources of error greatly depends on the choice of image points used for the computation of these angles, whereas the sensitivity of the computation of the relative tilt angle to the error due to image center position is only marginally affected by our choice of the image points. All of the analytical findings have been supported by extensive simulation.

4. THREE-DIMENSIONAL SHAPE AND OTHER TOPICS

Prof. Kanatani's research at Maryland also involved the study of three-dimensional shape properties and their behavior under camera transformations. This work is described in the following reports, the last of which is a monograph on the use of group representation theory in computer vision.

a) The Constraints on Images of Rectangular Polyhedra

This paper discusses how polyhedron interpretation techniques are simplified if the objects are rectangular trihedral polyhedra. This restriction enables one to compute the spatial orientation of a given corner and its motion from its image. The solution is expressed in terms of polar coordinates, Eulerian angles and quaternions. Then, based on the fact that the transformations mapping eight possible configurations of the rectangular corner to each other form a group isomorphic to $Z_2 \times Z_2 \times Z_2$, the corner configurations, their transformations, spatial orientations and states of face adjacency are expressed by triplets of binary bits, and the conditions constraining relationships among them are described in algebraic equations in terms of those triplets. Finally, the visibility conditions are formulated, and an algorithm of shape interpretation and hidden line detection from "local" information is presented. Some examples are given to compare our scheme with existing ones. The possible non-uniqueness of the interpretation and the effect of projective distortion are also discussed.

b) Shape from Texture: General Principle

The 3D shape of a textured surface is recovered from its projected image on the assumption that the texture is homogeneously distributed. First, the homogeneity of a discrete texture consisting of dots and line segments is defined in terms of the theory of distributions. Next, distortion of the observed texture density due to perspective projection is described in terms of the first fundamental form, which is expressed with respect to the image coordinate system. Based on this result, the basic equations to determine the surface shape are derived for both planar and curved surfaces, and numerical schemes are proposed to solve them. Necessary data are obtained in the form of summation or integration of functions over the texture elements on the image plane. Ambiguity in the interpretation of curved surfaces is also analyzed. Finally, numerical examples for synthetic data are presented, and our method is compared with other existing methods. It is shown that all other methods can be explained in terms of our formulation.

c) Constraints on Length and Angle

Given a perspective projection of line segments on the image plane, the constraints on their 3D positions and orientations are derived when their true length or the true angles they make are known. The line segments under consideration are first mapped to the center of the image plane as if the camera were rotated to aim at them. Then, the constraints are given by the geometry of perspective transformation, and the relations obtained are transformed back to the original configuration in the scene. An application is given to the interpretation of rectangular corners of polyhedra.

d) Camera Rotation Invariance of Image Characteristics

The image transformation due to camera rotation relative to a stationary scene is analyzed, and the associated transformation rules of "features" given by weighted averaging of the image are derived by considering infinitesimal generators on the basis of group representation theory. Three dimensional vectors and tensors are reduced to two dimensional invariants on the image plane from the viewpoint of projective geometry. Three dimensional invariants and camera rotation reconstruction are also discussed. The result is applied to the shape recognition problem when camera rotation is involved.

e) Group Theoretical Methods in Image Understanding

This work is a brief summary of mathematics, in particular group representation theory, relevant to the study of image understanding and computer vision. We introduce fundamentals of group representation theory, theory of invariants and theory of Lie groups and Lie algebra, especially representations and invariants of the 2D and 3D groups of rotations $SO(2)$, $SO(3)$, in relation to image understanding and computer vision.

Another project dealing with shape approximation is being carried out by Dr. Isaac Weiss, and is described in two reports:

f) 3-D Shape Representation by Contours

The question of 3-D shape representation is studied on the fundamental and general level. The two aspects of the problem, (i) the reconstruction of a 3-D shape from a given set of contours, and (ii) finding "natural" coordinates on a given surface, are treated by the same theory. We first state a few basic principles that should guide any shape reconstruction mechanism, regardless of its physical implementation. Second, we propose a new mathematical procedure that complies with these principles and offers several advantages over the existing *ad hoc* treatments. Some general results are derived from this procedure, which conform very well with human visual perception.

g) Curve Fitting with Optimized Mesh Point Placement

A recent theory of 3-D surface interpolation has been implemented numerically for the special case of curve fitting, in the plane and in 3-D space. The implementation demonstrates some of the major advantages of the theory, such as the ability of the curve to turn sharp corners, by concentrating mesh points (knots) in high curvature parts of the curve. This knot placement, which is a universal difficulty in previous interpolation or smoothing techniques, is done here automatically by the same principle of energy minimization that is used to fit the curve to the data.

A paper based on these reports appears in these Proceedings.

Dr. Weiss has also developed a new approach to the important problem of straight line fitting in a noisy image. The conventional least squared distance method of fitting a line to a set of data points is notoriously unreliable when the amount of random noise in the input (such as an image) is significant compared with the amount of data correlated to the line itself. Points which are far away from the line are usually just noise, but they contribute the most to the distance averaging, skewing the line from its correct position. A statistical method of separating the data of interest from random noise, based on a maximum likelihood principle, has been developed.

One final project deals with a model-driven, region-based approach to image segmentation; it has been applied to aerial photographs of houses and roads. A knowledge-based system for interpreting aerial photographs, Picture Query (PQ), first segments an image into primitive, homogeneous regions, then searches among combinations of these to find instances which satisfy definitions of object types. If primary evidence is insufficient, there may be an hypothesis-based search for the supporting evidence of related objects. This secondary search is restricted to windows by expected spatial relations. First instances are improved by searching for overlapping variants having better goodness-of-figure. The process may be repeated using re-estimated parameters of object definitions based on instances found previously. Results are reported for images of suburban neighborhoods, including roads, houses, and their shadows.

This paper appears in full in these Proceedings.

List of Technical Reports

1) Ken-ichi Kanatani, Analysis of Structure and Motion from Optical Flow, Part I: Orthographic Projection, CAR-TR-160, CS-TR-1576, October 1984, Revised June 1985.
2) Ken-ichi Kanatani, Analysis of Structure and Motion from Optical Flow, Part II: Central Projection, CAR-TR-161, CS-TR-1577, January 1985, Revised June 1985.
3) Ken-ichi Kanatani, Transformation of Optical Flow by Camera Rotation, CAR-TR-163, CS-TR-1580, November 1985.
4) Ken-ichi Kanatani, Structure from Motion without Correspondence: General Principle, CAR-TR-164, CS-TR-1581, November 1985.
5) Ken-ichi Kanatani, The Constraints on Images of Rectangular Polyhedra, CAR-TR-165, CS-TR-1582, November 1985.
6) Muralidhara Subbarao, Interpretation of Image Motion Fields: A Spatio-Temporal Approach, CAR-TR-167, CS-TR-1589, December 1985.
7) Eliahu Wasserstrom, Subpixel Registration, CAR-TR-173, CS-TR-1601, January 1986.
8) Fang-Jyh Lin Liu, Roger Eastman, and Larry S. Davis, Experiments in Stereo Matching using Multiresolution Local Support, CAR-TR-183, CS-TR-1617, January 1986.
9) Ken-ichi Kanatani and Tsai-Chia Chou, Shape from Texture: General Principle, CAR-TR-184, CS-TR-1618, February 1986.
10) Allen M. Waxman, Behrooz Kamgar-Parsi, and Muralidhara Subbarao, Closed-Form Solutions to Image Flow Equations for 3-D Structure and Motion, CAR-TR-190, CS-TR-1633, February 1986.
11) Jacqueline Le Moigne and Allen M. Waxman, Structured Light Patterns for Robot Mobility, CAR-TR-191, CS-TR-1634, February 1986.
12) Uma Kant Sharma and Larry S. Davis, Road Following by an Autonomous Vehicle Using Range Data, CAR-TR-194, CS-TR-1639, March 1986.

13) Behrooz Kamgar-Parsi, Practical Computation of Pan and Tilt Angles, CAR-TR-195, CS-TR-1640, March 1986.

14) Muralidhara Subbarao, Interpretation of Image Motion Fields: Rigid Curved Surfaces in Motion, CAR-TR-199, CS-TR-1654, April 1986.

15) Ken-ichi Kanatani, Constraints on Length and Angle, CAR-TR-200, CS-TR-1655, April 1986.

16) Ken-ichi Kanatani, Camera Rotation Invariance of Image Characteristics, CAR-TR-202, CS-TR-1663, May 1986.

17) Ken-ichi Kanatani and Tsai-Chia Chou, Tracing Finite Motions Without Correspondence, CAR-TR-211, CS-TR-1689, August 1986.

18) Ken-ichi Kanatani, Group Theoretical Methods in Image Understanding, CAR-TR-214, CS-TR-1692, August 1986.

19) John Aloimonos and Azriel Rosenfeld, Monocular Stereopsis: Theory and Applications, CAR-TR-218, CS-TR-1698, August 1986.

20) Muralidhara Subbarao, Interpretation of Visual Motion: A Computational Study, CAR-TR-221, CS-TR-1706, September 1986.

21) Isaac Weiss, 3-D Shape Representation by Contours, CAR-TR-222, CS-TR-1707, September 1986.

22) David Harwood, Susan Chang, and Larry S. Davis, Interpreting Aerial Photographs by Segmentation and Search, CAR-TR-223, CS-TR-1709, September 1986.

23) Isaac Weiss, Curve Fitting with Optimized Mesh Point Placement, CAR-TR-224, CS-TR-1710, September 1986.

24) Isaac Weiss, Straight Line Fitting in a Noisy Image, CAR-TR-234, CS-TR-1727, November 1986.

25) Behrooz Kamgar-Parsi, An Efficient Line Search Algorithm for Optimization of Multivariate Functions, CAR-TR-239, CS-TR-1738, November 1986.

26) Behrooz Kamgar-Parsi and Roger Eastman, Calibration of a Stereo System With Small Relative Angles, CAR-TR-240, CS-TR-1739, November 1986.

USC Image Understanding Research - 1986[1]

R. Nevatia

Institute for Robotics and Intelligent Systems
PHE 204 MC 0273
University of Southern California
Los Angeles, CA 90089-0273

ABSTRACT

This paper summarizes the USC Image Understanding research projects and provides references to more detailed projects and provides references to more detaile papers. Our work has focussed on the topics of: mapping from aerial images, robotics vision, motion analysis for ALV, some general techniques and parallel processing.

1 INTRODUCTION

Our image understanding research during 1986 has focussed on the following topics:

- Mapping from Aerial Images
- Robotics Vision
- Motion Detection for Autonomous Land Vehicle (ALV), and
- General Techniques
- Parallel Processing

In the following we summarize our research in these areas and provide references that contain more detail.

2 MAPPING FROM AERIAL IMAGES

Our goal here is to produce high-quality symbolic maps of complex, cultural scenes from aerial image data. As a test domain, we have chosen large commercial airport complexes. Such scenes have a variety of features such as the transportation network (runways, taxiways and roads), buildings (terminals, hangars, etc.) and mobile objects (airplanes, trucks, cars, etc.). Our aim is to produce descriptions of the individual objects in the scene as well as an integrated description of the entire scene including the functional relationships between the parts.

Initially, we have concentrated on extraction of runways. This is a task much more complex than might appear at first as runways are not smooth homogeneous patches but contain a variety of markings, tire-tread and oil spots, shoulders, patches of different material etc. In previous work, [1] we reported a technique that ***hypothesized*** the location of runways based on some evidence, even though the entire runway pieces were not necessarily visible. We have extended these methods to ***verify*** the generated hypotheses. The verification is primarily based on detecting the markings on a runway that are mandated by aviation standards and are also features heavily used by human pilots and hoto-interpreters. These techniques are described in another paper in these proceedings [2].

In separate work, supported by the Defense Mapping Agency (DMA), we are developing methods for detection and description of buildings. A method suitable for detection of relatively simple shapes is described in [3]. We are developing methods for handling more complex structures. These methods require more complex methods of organizing low-level features into meaningful higher level structures. Also, use of 3-D height data derived from stereo is being explored.

3 ROBOTICS VISION

Our concentration here is on description of 3-D shape. We are developing methods for both surface and volume descriptions. Both methods rely on some underlying phitosophical concepts - that complex shapes need to be described by decomposition into "simpler" parts and that the inter-relations of the parts are a significant aspect of the shape description. The decomposition can be carried out successively to the described level of detail. We call such descriptions ***structured, heirarchical descriptions.***

[1]This research was supported, in part, by the Defense Advanced Research Projects Agency contracts DACA76-85-C-0009 and F33615-84-K-1404, order No. 3119 and monitored by the U.S. Army Engineer Topographic Laboratories and Air Force Wright Aeronautical Laboratories respectively.

For surfaces, we believe that appropriate places for segmentation are at the occluding (or jump) boundaries, slope discontinuities (folds) and "ridges". These features can, in turn, be computed from the curvature properties of surfaces and correspond to points or lines where the curvature is a local extremum or goes through a zero-crossing. Actual computation of these properties is made difficult due to the presence of noise in digital range data. Previously, we had reported a method for extracting these significant features from range data [4]. We have extended these methods so that we are now able to get complete surface segmentation and a description of their relationships, see [5].

For volume descriptions, we use the generalized cone representation. However, this time we assume that we have only sparse and imperfect 3-D data available, as is typically the case if range is obtained by stereo analysis. Our technique currently works on the class of objects known as *linear, straight, homogeneous generalized cylinders* (LSHGCs) and we have demonstrated this method on stereo analysis of real scenes (described in [6]. We are in the process of extending the techniques to more general classes of generalized cones.

4 MOTION DETECTION AND ANALYSIS

Our motion research primarily addresses the detection of moving objects and the description of the motion of the objects with the derivation of the description of the object being a secondary concern. Within the ALV task domain, the simultaneous motion of the camera and objects in the scene further complicates the problem.

Of the many alternative approaches to motion analysis, we have chosen the long range or feature point methods. This approach involves extracting a set of reliable features in a sequence of images (lines, corners, contours, regions, etc.), finding the corresponding features in the sequence (i.e. by a series of image to image matching operations), and finally the computation of three-dimensional motion estimates based on the series of correspondences. We have addressed each of the problems separately and have begun to combine them into a coherent system.

The region based segmentation and matching method come directly from older techniques [7,8]. Complete results of this relaxation based graph matching technique applied to the motion problem are given in another paper [9] . The histogram segmentation attempts to find compact regions, relatively uniform in intensity.

Properties of the regions and relations between them are used by the matching program to find corresponding regions in consecutive images. Neither of these operations depends explicitly on the fact that the images are from a motion sequence (segmentations are performed independently, matching is a series of pair-wise comparisons). Still, this technique was adequate for producing data for the motion estimation program, with variations in the segmentation causing the most problems.

The second feature based method uses contours (connected edges from a zero-crossing edge detector). This method grew out of earlier work in segment matching, but extended to connected curves and using constraints from the motion task. This method starts with a straight line segment description of the edges in the images and finds corresponding line segments. The initial segment matches guide a contour matcher which uses local shape similarity, followed by a more global (relaxation based) matching step to choose which of the possible matches is most likely. This procedure works very well with small motions and is stable enough to track the features through several frames.

Both of these methods are intended for use by our general three-dimensional motion estimation program. Rather than considering only two frames, we chose to consider longer sequences and to treat the sequence as a whole rather than a set of pairs. This resulted in a simplification of the computation of the motion estimate and a relatively robust algorithm. The equations were derived for several cases: 3 points in 3 frames, 2 points in 4 frames, and 1 point in 5 frames, with the last one being implemented on the Symbolics and tested. Pure translation is much simpler and was also implemented for 1 point in 3 frames. This method has been effective computing reasonable motion estimates along with relative depths for the points. A set of results are given in another paper [9] for both a synthetic case and two examples from real sequences.

This work continues for the contour matching method and in using motion estimates to constrain the matching for later positions of the the sequence.

5 GENERAL TECHNIQUES

This term refers to the class of techniques that are needed for a variety of application tasks. The techniques we have studied recently are:

1. Fast Convolution for edge detection
   Convolution is often the most expensive step in the edge detection, especially when large masks are used. In the case of the rotationally invariant Laplacian-of-Gaussian filter (LoG), we have developed a procedure which takes advantage of the spectral characteristics of the filter to speed up the computation. Basically,

since the LoG is a bad pass filter, we may fold the spectrum of the image (after low-pass filtering) without loss of information, which is equivalent to reducing the resolution. We first convolve the image with a Gaussian filter, then reduce the resolution by decimating this convolved image, and convolve this reduced image with a small LoG mask. To recover the full resolution image, we use a simple bilinear interpolation scheme. All parameters can be obtained automatically for any value of σ. The paradoxical result is that the computation time decreases as σ increases.

2. Estimation and accurate localization of edges.
   Edge detection if of crucial importance to image understanding, and accurate localization is needed by many higher level processes such as stereo. Laplacian-of-Gaussian filters (LoG), either directional or rotationally symmetric have been advanced as choice candidates for edge detection. It has been noted, however, that they introduce a bias in edge location when either the edge profile is not symmetric or when nearby edges interact. We have investigated both sources of error and developed methods to correct this bias (in 1-D).

   (a) Bias due to the edge profile
   We propose different methods to correct the bias of the LoG operators: the first one combines the zero-crossing information at 2 different resolutions whereas the others use a single resolution but sample the convolved output at more locations besides the zero-crossings.

      i. 2 resolutions
      By solving a system of 2 linear equations, we can recover the true location of an edge. We can also estimate the step height and the slopes difference for each edge. The drawback of this method is that we have to establish correspondences between zero-crossings at 2 different resolutions.

      ii. Using a single resolution
      Here we can recover the true location of an edge by locating either the positive and negative extrema associated with a zero-crossing, or by finding the convolutioin output at arbitrary locations close to the zero crossing. Here again, we find not only the location of the edge with subpixel precision, but also the step height and slopes difference.

   (b) Bias due to edge interaction
   We developed a method to detect, locate and estimate edge in a 1-D signal, which explicitly models and corrects interactions between nearby edges. The method is iterative with initial estimation of edges provided by the zero crossings of the signal convolved with LoG mask: We model edges as perfect steps, specified by their location and magnitude. As a result, the convolution of the signal with LoG filters can be expressed at any point by a weighted sum of the responses from each discontinuity.

   During the initialization phase, the number of discontinuities is given by the number of zero crossings, the location by the (interpolated) position of the zero crossings, and the magnitude by an estimation involving convolution of the signal with the first derivative of a Gaussian, as explained in detail in the next subsection.

   The iteration proceeds by hypothesize and verify : Supposing that the signal is well approximated by the initial train of discontinuities, we can synthesize the corresponding convolution with a **LoG** mask. Due to the interaction between nearby edges, the correct location of a discontinuity is no longer at the zero crossing position but at some nearby location corresponding to a nonzero level crossing. Once the new locations are computed, we estimate the corresponding magnitude of the step. This iterative improvement continues until all values have converged to a stable position, or until the improvement become negligible.

   It is important to notice that we only use the values of the signal convolved with the **LoG** filter in a small neighborhood around each zero crossing to perform our computation. It is also interesting to note that, even though our model is very crude, it estimates the edges so nicely that the reconstructed signal is a very acceptable approximation of the original.

3. Stereo Analysis - Several stereo systems have been developed in recent years and they generally exhibit good performance. However, all are susceptible to errors in complex scenes, particularly at surface discontinuities. We are developing techniques that rely on more global analysis of the scene than just individual edges or line segments. Our approach and some preliminary results are described in another paper in these proceedings [10].

6 PARALLEL PROCESSING

As vision systems start becoming more practical, we need to be concerned with the speed of execution. It is clear that the needed speed can only be obtained by the use of highly parallel machines. The use of parallel machines for the lower levels (the iconic levels) of processing is rather straight-forward, but not so for the higher levels.

Techniques of our group rely heavily on the extraction and use of symbolic descriptions and such techniques are much more difficult to parallelize. However, we believe that we have made a good start in understanding how to implement some of our key algorithms, these efforts are described in [11].

We have also been studying parallel processing of production systems. This project is a continuation of our previous work. In the last year the following three results were obtained: developed a technique for reducing the production system search space, developed a technique for mapping production system into fixed size multiprocessors by partitioning the problem and performed preliminary design for a specialized multiprocessor for production systems. The reduction of search spaces is important because it increases the speed for achieveing the solution. By studying the interdependencies between rules, we were able to eliminate redundant paths in the search paper. The mapping of production systems into multiprocessor is also based on the analysis of rule interdependencies. The design of a 16-node multiprocessor has started, and we have built one node. In this problem, rules can fire parallel and rules communicate via message-passing technique.

References

[1] R. Nevatia. Image understanding research at usc: 1984-85. In *Proceedings of the DARPA Image Understanding Workshop*, Miami, Fla., December 1985.

[2] A. Huertas, B. Cole, and R. Nevatia. Detecting runways in aerial images. In *Proceedings of the DARPA Image Understanding Workshop*, Los Angeles, California, February 1987.

[3] R. Nevatia and A. Huertas. Building detection in simple scenes. In *USC-ISG Report 110*, September 1985.

[4] T.J. Fan, G. Medioni, and R. Nevatia. Description of surfaces from range data. In *Proceedings of the DARPA Image Understanding Workshop*, pages 232–244, Miami Beach, Florida, December 1985.

[5] T.J. Fan, G. Medioni, and R. Nevatia. Surface segmentation and description from curvature features. In *Proceedings of the DARPA Image Understanding Workshop*, Los Angeles, California, February 1987.

[6] G. Kashipati and R. Nevatia. From sparse 3-d data *Directly* to volumetric shape descriptions. In *Proceedings of the DARPA Image Understanding Workshop*, Los Angeles, California, February 1987.

[7] R. Ohlander, K. Price, and R. Reddy. Picture segmentation by a recursive region splitting method. *Computer Graphics and Image Processing*, 8:313–333, 1978.

[8] O.D. Faugeras and K. Price. Symbolic description of aerial images using stochastic labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(6):633–642, November 1981.

[9] H. Shariat and K. Price. Results of motion estimation with more than two frames. In *Proceedings of the DARPA Image Understanding Workshop*, Los Angeles, California, 1987.

[10] S. Cochran. Steps toward accurate stereo correspondence. In *Proceedings of the DARPA Image Understanding Workshop*, Los Angeles, California, February 1987.

[11] D. Reisis and V.K.P. Kumar. Parallel processing of image and stereo matching using linear segments. In *IRIS Report 204*, Los Angeles, California, January 1987.

IMAGE UNDERSTANDING RESEARCH AT SRI INTERNATIONAL

Martin A. Fischler and Robert C. Bolles

SRI International (Artificial Intelligence Center)
333 Ravenswood Avenue, Menlo Park, California 94025

Abstract

The SRI Image Understanding program is a broad effort spanning the entire range of machine vision research. In this report we describe the progress in two programs. The first is concerned with modeling the earth's surface from aerial photographs; the second is concerned with visual interpretation for land navigation. In particular, we describe progress in (1) the design of a core knowledge structure; (2) representing, recognizing, and rendering complex natural and man-made objects; (3) recognizing and modeling terrain features and man-made objects in image sequences; (4) interactive techniques for scene modeling and scene generation; (5) automated detection and delineation of cultural objects in aerial imagery; and (6) automated terrain modeling from aerial imagery.

1 INTRODUCTION

The overall goal of the SRI Image Understanding research program is to obtain solutions to fundamental problems in computer vision that are necessary to allow machines to model, manipulate, and understand their environment from sensor-acquired data and stored knowledge.

In this report we describe progress in two programs.[1] The first is concerned with modeling the earth's surface from aerial photographs; the second is concerned with allowing a robotic device to successfully navigate through, and interact with, a natural 3-D environment based on real-time interpretation of sensory data.

In the first program we describe our progress in developing techniques for: (a) automated terrain modeling from aerial imagery; (b) automated detection and delineation of cultural objects in aerial imagery; and (c) interactive techniques for scene modeling and scene generation.

In the second program we describe progress in developing techniques for automated real-time recognition of terrain features and man-made objects from image sequences acquired by a combination of ranging and photographic sensors.

Common to both programs, we describe progress in: (a) developing new techniques for representing, recognizing, and rendering complex natural and man-made objects; and (b) the construction of a core knowledge structure (CKS), which can serve as the integrating mechanism for a new generation of generic vision systems. These systems will be knowledge-base driven, rather than task-specific using techniques in which domain knowledge is compiled into the interpretative algorithms.

[1]Supported by the Defense Mapping Agency and Defense Advanced Research Projects Agency under contracts MDA903-86-C-0084 and DACA76-85-C-0004.

2 DESIGN OF A CORE KNOWLEDGE STRUCTURE

The natural outdoor environment imposes significant obstacles to the design and successful integration of the interpretation, planning, navigational, and control functions of a general purpose vision system. Many of these functions cannot yet be performed at a level of competence and reliability necessary to satisfy the needs of an autonomous robotic device. Part of the problem lies in the inability of available techniques, especially those involved in sensory interpretation, to use contextual information and stored knowledge in recognizing objects and environmental features. Our goal in this effort, described in greater detail in a separate paper (G.B. Smith and T.M. Strat, "Information Management in a Sensor-Based Autonomous System," in these proceedings), is to design a core knowledge structure that can support a new generation of knowledge-based generic vision systems.

A key scientific problem we address in this task is to devise a way of describing the appearance and characteristics of any given physical environment so thoroughly that we are assured that any deficiencies in available perception (vision) techniques can be overcome by access to such prior knowledge. We cannot resort to the equivalent of using a pixel-level description as the only or ultimate solution because such detailed data would be impractical to obtain, store, retrieve, or use in the interpretive process; such low-level data would almost certainly be inaccurate when obtained, and would quickly degrade as physical changes occur. (Even illumination changes would cause a pixel level description of appearance to become useless.) Finally, accurate interpretation must be based on more than just image appearance, and there is no immediately obvious way of describing and storing such semantic (nonpictorial) information at arbitrary levels of detail.

The CKS is designed as a community of independent interacting processes that cooperate in achieving the goals of the scene modeling system. These processes may represent sensors, interpreters, controllers, user interface drivers, or any other information processor. Each process can be both a producer and a consumer of information. Each has access to, and control over, a certain limited portion of the knowledge/database resources. The CKS architecture permits access to stored knowledge by both geographic location and by semantic content.

A semantic directory provides a means of access to the data that is orthogonal to that provided by the more conventional spatial directory. While the spatial directory allows rapid retrieval of relevant data based upon spatial location, the semantic directory provides access to the same data based upon the semantic attributes of those data. Like the spatial directory, it supports description at multiple levels of resolution — concepts are described and data are retrieved at the level of abstraction that is most appropriate for the purpose at hand. The semantic directory also serves as a blackboard whereby one computational process can communicate with another in terms of semantic descriptions about which it is knowledgeable.

The implementation of the semantic directory consists of two components: a vocabulary of terms and a set of connections among them that serve to define their inherent relationships and properties. The vocabulary consists of a carefully chosen, domain-specific set of terms that have been identified as being both useful for, and instantiable by, the computational processes. Relationships and properties are specified by several methods. A semantic network is used to encode the specialization lattice of the concepts and the physical decomposition of composite objects. Procedural definitions are provided when it is necessary to do so. A relational database is used to retain semantic relationships that are explicitly provided. These relationships allow class properties to be inherited and alternative semantic classes to be enumerated. Each process that interacts with this knowledge base need only know how to translate to and from this vocabulary in order to share information with any other process. This avoids the need for direct translation of data between the knowledge representation of every pair of independent processes. The key motivation for this approach is to facilitate the integration of independent processes and subprocesses without restricting the representations they may use to encode their knowledge.

In conjunction with our development of the core knowledge structure (CKS), we have selected four tasks to demonstrate the adequacy and effectiveness of [illegible] CKS design. These tasks are object recognition, scene rendering, mission planning, and dynamic data prediction.

An initial implementation of the complete CKS concept will be demonstrable by March 1987. A human operator will be able to interact with the system through a sophisticated 2-D sketch map interface or through a 3-D display capability (probably based on SuperSketch). Complete integration with both a semantic labeling module and with a range-data analysis module will serve to demonstrate interaction with sensory processes. Some metalevel processes will have been developed to perform resource allocation and control functions; vocabulary and semantic networks will have been implemented to allow description of a significant ALV-type scenario. Our future plans are to make the CKS the functional core of an ALV-type computer vision demonstration system. However, the ultimate success of this system will be shaped by our ability to resolve some of the still open scientific questions — especially extensions to the scope and power of the semantic vocabulary, and evolution of our current ideas about how to deal with the problems of multisource integration and conflict resolution.

3 REPRESENTING, RECOGNIZING, AND RENDERING COMPLEX NATURAL AND MAN-MADE OBJECTS

The main theoretical issue we address in this task is how to model a large class of natural and man-made objects in a functionally useful way. Our approach is to develop a representation system that facilitates object recognition by providing the information necessary to predict such things as where an object is likely to be, what it is expected to look like, and how much of it is likely to be visible. Our recognition strategy is to predict as much as possible from the task description, and then use the predictions to reduce the search space to be analyzed for a match. In some cases, such as the recognition of a known landmark, these predictions may be quite specific, even down to the level of a template of range values. In other cases, such as the recognition of a generic object like a person, the predictions may be at a much more abstract level, such as a configuration of parts. The type of predictions produced dictates, to a large extent, the recognition process.

We are developing a unified representation system that is built on the SuperSketch system developed by Alex Pentland at SRI ["Perceptual Organization and the Representation of Natural Form," 1986]. It uses a set of shape and surface texture primitives and a set of rules for combining and transforming these primitives. The primitives include superquadrics and fractals, in addition to more conventional models (e.g., splines). Superquadrics are well-suited for describing natural objects because they cover a continuous range of shapes from ellipsoids to cubes. Fractals provide a concise way of modeling statistical structures, such as mountainous terrain or rough surfaces. Superquadric primitive shapes can be stretched or bent by a wide variety of transformation rules, and then combined to form complex objects, such as a tree consisting of a fluted trunk with an irregular crown. The ability of a single modeling scheme to compactly represent a variety of shapes that can be incrementally changed is crucial in the ground-level navigation domain, because objects that are viewed over time appear to change shape and gradually become more distinct as they are approached.

Given a SuperSketch model of a scene, we have developed techniques to render it in several different ways. In addition to the conventional three-dimensional graphics techniques, we have added range prediction techniques to produce synthetic depth images and fractal methods to produce realistic surface textures for such things as mountains, bushes, and trees. We also produce an image, which we call the "paint-by-numbers" image, that identifies the model component visible at each pixel. This image is particularly useful for estimating which components are likely to be visible and to what extent.

As a demonstration of the power of this prediction approach, we have combined a representation of uncertainty with the SuperSketch model and used the pair to produce weighted templates for recognizing known three-dimensional objects in range data. We use joint-normal distributions to represent the locational uncertainties and standard techniques for combining the distributions [Barnard, 1986]. To recognize an object, such as a gate with a known structure, we combine the uncertainty associated with the vehicle's location with the uncertainty of the gate's location, and then map the resulting distribution into the range image in order to predict the region in which the gate is likely to appear. Then we predict a template of range values for a prominent piece of the gate and search the region for a match to the template. This simple technique is quite effective; however, it is only appropriate for tightly constrainted tasks.

For less structured tasks we are developing a technique that segments a range image into a set of parts (i.e., superquadric volume primitives), which we intend to use as the input to a higher-level matching procedure [Pentland, "Recognition by Parts," Dec. 1986]. This technique segments the raw range data into coherent units and describes each unit as a superquadric. Our initial implementation has produced some promising results. However, it is computationally expensive and requires some interactive preprocessing. The potential benefit of such a technique is that it could produce high-level descriptions of a scene which would significantly reduce the search required to recognize complex objects.

In the future we plan to extend the segmentation system that characterizes a scene as a set of overlapping superquadrics, develop new ways to refine object descriptions continuously as additional data are acquired, and explore recognition techniques that base their decisions on combined color properties and range-derived geometric features.

4 RECOGNITION AND MODELING OF TERRAIN FEATURES AND MAN-MADE OBJECTS FROM IMAGE SEQUENCES

Our goal in this research task is to develop automated methods for producing a labeled three-dimensional scene model from many images recorded from different viewpoints and from image sequences. We view the image sequence approach as an important way to avoid many of the problems that hamper conventional stereo techniques because it provides the machine with more information about the scene. The "redundant" information can be used to increase the precision of the data or filter out artifacts; The new information provided by the additional images can help disambiguate matches for features occurring along occlusion boundaries and in the midst of periodic structures.

We are developing two techniques for building three-dimensional descriptions from multiple images. One is a range-based technique that builds scene models from a sequence of range images, and the second is a motion analysis technique that analyzes long sequences of intensity images. The range technique uses data from an inertial guidance sensor on the vehicle to compenstate for vehicle attitude and position changes caused by bumps, curves, and speed changes. As a result the range data are transformed into a static world coordinate system, which is a necessary first step for almost all further analysis. By combining the data from multiple images, we are able to filter out artifacts and produce a more complete map of the region in front of the vehicle. We have developed several representations of this three-dimensional data, including height maps, orientation images, and voxel arrays, each of which offers distinct coherence and resolution advantages to the analysis procedures.

We are developing a set of techniques that analyze these representations of the range data to identify key navigational features, such as support surfaces, ditches, bushes, and thin raised objects. The technique to identify support surfaces examines the portion of the "terrain map" in front of the vehicle for large regions of relatively smooth patches. The ditch detector also analyzes the terrain map, but it looks for a sequence of depressions that exceed a predetermined depth. The bush detector and thin raised object detector examine portions of the range data that are above the support surface. The annealing-based technique described in the previous section is used to build descriptions of bushes. The thin raised object detector concentrates on the raw range images because they contain the best continuity information for thin things like fence posts and telephone poles. The problem with thin things is that the beam of the range sensor often hits other surface in addition to the object of interest, which produces "mixed" range values.

In [Bolles, Baker, & Marimont, "Epipolar-Plane Image Analysis: An Approach to Determining Structure from Motion," to appear in the International Journal of Computer Vision, winter of 1987] we describe a motion analysis technique, which we call Epipolar-Plane Image (EPI) Analysis. It is based on considering a dense sequence of images as forming a solid block of data. Slices through this solid at appropriately chosen angles intermix time and spatial data in such a way as to simplify the partitioning problem: these slices have more explicit structure than the conventional images from which they were obtained.

We have demonstrated the feasibility of this novel approach for depth analysis, however, the initial implementation has significant limitations. One of them is due to the fact that we analyze one slice at a time. The problem is that by concentrating on an individual slice we ignore one spatial dimension. Therefore, we are currently finishing a second implementation of this technique that maintains continuity in all three dimensions by the forming and analyzing three-dimensional surfaces in the spatio-temporal volume. The second spatial dimension provides us with a way to maintain object conerence between features observed on separate epipolar planes. This new implementation also provides a way to perform the analysis incrementally as new images are received and a way to analyze sequences gathered by a sensor that is changing its view direction as it moves along a straight path.

5 INTERACTIVE TECHNIQUES FOR SCENE MODELING AND SCENE GENERATION

Manual photointerpretation is a difficult and time-consuming step in the compilation of cartographic information. On the other hand, fully automated techniques for this purpose are currently incapable of matching the human's ability to employ background knowledge, common sense, and reasoning in the image interpretation task. Near-term solutions to computer-based cartography must include both i[illegible]ractive extraction techniques, and new ways of using compu[illegible] technology to provide the end-user with useful information in a primarily iconic, rather than symbolic, format.

A summary of our progress in this area is described in a separate paper in these proceedings: "Design of a Prototype Environment," by A.J. Hanson, A.P. Pentland, and L.H. Quam. This paper describes an interactive modeling system appropriate for cartographic tasks. The system, already in an advanced state of development, permits entry and display of cartographic features registered to geographic coordinates, can interact with existing cartographic data bases, and supports automated and semiautomated feature compilation. An important capability of this system is its ability to provide an end-user with an interactively-controlled scene viewing capability that could eliminate the need to produce hard-copy (symbolic) maps in many application contexts.

6 AUTOMATED DETECTION AND DELINEATION OF CULTURAL OBJECTS IN AERIAL IMAGERY

The detection, delineation, and recognition, in aerial imagery, of any significantly broad class of objects (e.g., buildings, airports, cultivated land) has proven to be an extremely difficult problem. In fact, a nominal component in the solution of this problem, image partitioning, is considered to be one of the most refractory problems in machine vision. We are pursuing a research strategy for solving the partitioning and delineation problems involving the following steps:

1. Develop a high performance syntactic (i.e., no use of semantic or contextual knowledge) partitioning algorithm. This has largely been accomplished [Laws, 1985; Fua & Leclerc, 1987].

2. Starting with a "good" syntactically partitioned image, introduce generic semantic and geometric knowledge about a specific object class to detect and delineate instances of these objects. A number of referenced papers by Fua and Hanson, and a new paper in this proceedings [Fua & Hanson, 1987] describe our progress in this task.

As discussed in Fua and Hanson, our approach to delineating instances of complex generic shape models in aerial imagery is to reason from a combination of geometric and photometric cues. We avoid the use of templates by modeling generic cultural objects as networks of rectilinear structures enclosing areas of slowly varying intensity; linear features are modeled as networks of parallel curves, while vegetation clumps are modeled as consistently-textured areas enclosed by very jagged or fractal edge segments.

The expected locations of missing shape components are predicted and tested, resulting in rejection, or completion of hypothesized area-enclosing contours. Adaptive search methods are used to locate missing edges as well as to adjust the locations of known edges. The method used to analyze the photometric characteristics of an area uses a RANSAC approach [Fischler & Bolles, 1981] to compensate for, and identify, gross anomalies. While the shape descriptions we produce have immediate utility, they are also appropriate for input into semantics-driven application systems, and thus bridge a previously unfilled gap between low-level and high-level image understanding approaches.

The system can detect complex rectilinear cultural objects in a selection of monochrome, monocular aerial imagery; more work is necessary to stabilize the system across images and to improve the performance associated with the recently-developed nonrectilinear object models.

Current work has the objective of producing a concise representation for the models and their parsing rules that is easily altered and can be extended to handle contextual knowledge. Various ways of incorporating parts of this system into the SRI scene modeling system (see Hanson, Pentland & Quam, these proceedings) are also being investigated.

7 AUTOMATED TERRAIN MODELING FROM AERIAL IMAGERY

Stereo reconstruction is a critical task in cartography that has received a great deal of attention in the image understanding community. Its importance goes beyond its obvious application to constructing geometric models: understanding scene geometry is necessary for effective feature extraction and other scene analysis tasks. While considerable success has been achieved in important parts of the problem, there is no complete stereo mapping system that can perform reliably in a wide variety of scene domains.

The standard approach to the problem of stereo mapping involves finding pairs of corresponding scene points in two images (which depict the scene from different spatial locations) and using triangulation to determine scene depth. Various factors associated with viewing conditions and scene content can cause the matching process to fail; these factors include occlusion, projective or imagining distortion, featureless areas, and repeated or periodic scene structures. Some of these problems can only be solved by providing the machine with more information, which may take the form of additional images or descriptions of the global context.

Our research strategy in this task is to develop new techniques for the key steps in the stereo process, such as matching and interpolation, and in parallel, to investigate ways to integrate these new ideas with existing techniques. As part of this process we have implemented and evaluated a complete high-performance stereo system [Hannah, 1985].

We are currently investigating two novel approaches to stereo depth recovery, which are significant departures from the conventional paradigm and which have important implications for other problems in scene analysis.

Barnard [1986, 1987] embeds local matching, at the level of individual pixels, in a global optimization framework. His objective function rewards correspondences between pixels that are similar in intensity value and imply disparities that are similar to those of their neighboring pixels. Simulated annealing is used to find a complete set of correspondences that best satisfies the objective function. Because individual pixels (rather than finite areas) are matched, projective distortion is no longer a problem, nor does one have to worry about adjusting the size or shape of a correlation patch. Experiments show that this approach can successfully compile a dense model of natural 3-D (ground level) scenes.

G.B. Smith [1985, 1986] describes a method for dense stereo compilation that entirely avoids local matching. The procedure begins with stereo images assumed to be in correspondence so that depth recovery can be accomplished for individual scan lines (i.e., the horizontal scan lines in the two images are corresponding epipolar lines). Smith shows how to set up systems of simultaneous equations whose solution is the depth profile corresponding to the intersection of the epipolar plane with the scene surfaces. Experiments with synthetically generated scenes show that the technique is theoretically sound, but the approach requires further work because in its present implementation it is overly sensitive to noise.

While the stereo problem will remain a focus of a portion of our research effort, our main concern is to develop an understanding of how knowledge of scene depth information can be effectively employed in the scene partitioning and object recognition tasks.

8 ACKNOWLEDGEMENT

The following researchers have contributed to the work described in this report: H.H. Baker, S.T. Barnard, R.C. Bolles, O. Firschein, M.A. Fischler, P. Fua, M.J. Hannah, A.J. Hanson, K.I. Laws, Y. Leclerc, D. Marimont, A.P. Pentland, L.H. Quam, G.B. Smith, T.M. Strat, L. Wesley, and H.C. Wolf.

9 REFERENCES

Recent publications, resulting fully or in part from our image understanding research program, include:

1. Baker, H.H., R.C. Bolles, and D.H. Marimont, "Generalizing Epipolar-Plane Image Analysis for Non-Orthogonal and Varying View Directions," Proc. Image Understanding Workshop, U. of Southern California, *(these proceedings)*, 1987.

2. Barnard, S.T., "Stereo Matching by Hierarchical, Microcanonical Annealing," Proc. Image Understanding Workshop, U. of Southern California, *(these proceedings)*, 1987.

3. Barnard, S.T., R.C. Bolles, D. Marimont, and A.P. Pentland, "Multiple Representations for Mobile Robot Vision," Proc. SPIE Symposium on Advances in Intelligent Robotics Systems, Cambridge, Massachusetts, October 26-31, 1986.

4. Barnard, S.T., "A Stochastic Approach to Stereo Vision," Proc. 5th National Conf. on Artificial Intelligence, Phildelphia, Penna, pp. 676-680, August 11-15, 1986.

5. Bolles, R.C., H.H. Baker, and D.H. Marimont, "Epipolar-Plane Image Analysis: An Approach to Determining Structure from Motion," International Journal of Computer Vision, to appear, winter of 1987.

6. Bolles, R.C. and H.H. Baker, "Epipolar Image Analysis: A Technique for Analyzing Motion Sequences," Proc. 3rd. Int. Symp. on Robotics Research, Paris, France, October 1985.

7. Bolles, R.C., H.H. Baker, and D.H. Marimont, "Epipolar-Plane Image analysis: an Approach to Determining Structure from Motion," International Journal of Computer Vision, to appear, Jan. 1987.

8. Fischler, M.A. and R.C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," Comm. ACM, Vol.24(6), pp.381–395, June 1981.

9. Fischler, M.A. and O. Firschein, "Intelligence: the Eye, the Brain, and the Computer," Addison-Wesley, Reading, Massachusetts, 1987.

10. Fischler, M.A. and O. Firschein, "Readings in Computer Vision," Morgan-Kaufmann (in press; 1987).

11. Fischler, M.A. and O. Firschein, "Parallel Guessing: A Strategy for High Speed Computation," Pattern Recognition (in press, 1987).

12. Fua, P. and A.J. Hanson, "Using Generic Geometric Models for Intelligent Shape Extraction," Proc. Image Understanding Workshop, U. of Southern California, *(these proceedings)*, 1987.

13. Fua, P. and Y. Leclerc, "Finding Object Boundaries Using Guided Gradient Ascent," Proc. Image Understanding Workshop, U. of Southern California, *(these proceedings)*, 1987.

14. Fua, P. and A.J. Hanson, "Locating Cultural Regions in Aerial Imagery using Geometric Cues," Proc. DARPA Image Understanding Workshop, Miami Beach, Florida, pp. 271-278, Dec. 9-10, 1985.

15. Fua, P. and A.J. Hanson, "Using Generic Geometric Knowledge to Delineate Cultural objects in Aerial Imagery," Tech. Note 378, Artificial Intelligence Center, SRI International, Menlo Park, California, March 1986.

16. Fua, P. and A.J. Hanson, Resegmentation Using Generic Shape: Locating General Cultural objects," Pattern Recognition Letters, in press, 1986.

17. Hannah, M.J., "Evaluation of STEREOSYS vs other Stereo Systems," Technical Note 365, Artificial Intelligence Center, SRI international, Menlo Park, California, Oct. 1985.

18. Hannah, M.J., "The Stereo Challenge Data Base," Technical Note 366, Artificial Intelligence Center, SRI International, Menlo Park, California, Oct. 1985.

19. Hannah, M.J., "SRI's Baseline Stereo System," Proc. DARPA Image Understanding Workshop, Miami Beach, Florida, pp. 149-155, Dec. 9-10, 1985.

20. Hanson, A.J., A.P. Pentland, and L.H. Quam, "Design of a Prototype Interactive Cartographic Display and Analysis Environment," Proc. Image Understanding Workshop, U. of Southern California, *(these proceedings)*, 1987.

21. Laws, K.I., "Goal-Directed Textured-Image Segmentation," Proc. SPIE Conf. on Applications of Artificial Intelligence II," Vol. 548, Arlington, Virginia, April 9-11, 1985.

22. Laws, K.I, "Experiments in Navigational Road Tracking," in S.J. Rosenschein, M.A. Fischler, and L.P. Kaebling, "Research on Intelligent Mobile Robots, Final Technical Report, Project 7390, SRI International, Menlo Park, California, pp. 151-157, May 20, 1986.

23. Leclerc, Y., "Capturing the Local Structure of Image Discontinuities in Two Dimensions, " Proc. IEEE Conf. on Computer Vision and Pattern Recognition, San Francisco, California, pp. 34-38, June 19-23, 1985.

24. Marimont, D.H., "Projective Duality and the Analysis of Image Sequences," Proc. of the Workshop on Motion: Representation and Analysis, IEEE Computer Society, pp. 7-14, Kiawah Island, South Carolina, May 1986.

25. Pentland, A.P., "Fractal-Based Description of Natural Scenes, IEEE PAMI 6(6):661-674, 1984.

26. Pentland, A.P., "A New Sense for Depth of Field," Proc. 9th Int. Joint conf. on Artificial Intelligence, Los Angeles, California, pp. 988-994, August 18-23, 1985.

27. Pentland, A.P. (ed.), "From Pixels to Predicates," Ablex, Norwood, New Jersey, 1985.

28. Pentland, A.P., "On Describing Complex Surfaces," Image and Vision Computing, 3(4):153-162, November 1985.

29. Pentland, A.P., "Part Models," Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Miami Beach, Florida, pp. 243-249, June 22-26, 1986.

30. Pentland, A.P., "Parts: Structured Descriptions of Shape," Proc. 5th National Conf. on Artificial Intelligence, Philadelphia, Pennsylvania, pp. 695-701, August 11-15, 1986.

31. Pentland, A.P., "Perceptual Organization and Representation of Natural Form," Artificial Intelligence, 28:293-331, 1986.

32. Pentland, A.P., "Recognition by Parts," SRI Technical Note 406, Dec. 1986.

33. Quam, L.H., "The Terrain-Calc System," Proc. DARPA Image Understanding Workshop, Miami Beach, Florida, pp. 327-330, Dec. 9-10, 1985.

34. Smith, G.B. and T.M. Strat, "Information Management in a Sensor-Based Autonomous System," Proc. Image Understanding Workshop, U. of Southern California, *(these proceedings)*, 1987.

35. Smith, G.B., "Stereo Reconstruction of Scene Depth," Proc. Computer Vision and Pattern Recognition, San Francisco, California, pp. 271-276, June 19-23, 1985.

36. Smith, G.B., "Stereo Integral Equation," Proc. 5th National Conf. on Artificial Intelligence, Phildelphia, Pennsylvania, pp. 689-694, August 11-15, 1986.

37. Strat, T.M. and M.A. Fischler, "One-Eyed Stereo: A General Approach to Modeling 3-D Scene Geometry,", Proc. 9th Int. Joint Conf. on Artificial Intelligence, Los Angeles, California, pp. 937-943, August 18-23, 1985.

38. Wesley, L.P., "Evidential Knowledge-Based Computer Vision," Optical Engineering, 25(3):363-379, March, 1986.

IMAGE UNDERSTANDING: INTELLIGENT SYSTEMS *

Thomas O. Binford

Artificial Intelligence Laboratory, Computer Science Department,
Stanford University, Stanford, CA 94305, U.S.A.

Abstract

This report summarizes progress in intelligent systems for Image Understanding. Recent research has greatly expanded the class of geometric models used in our new intelligent system and has extended the scope of symbolic methods in using geometric models. [Ponce 87a] describes analytic solutions for projections of a large class of generalized cylinders, used in prediction and display. They show invariants under projection which have been incorporated in a program for estimating the axes of generalized cylinders from their images. [Ponce 87b] presents an iterative method for computing intersections of complex geometric models. [Lowry 87a] describes algebraic methods in automatic programming for intelligent vision systems; these methods have been applied to linear programming algorithms. These methods are part of a broad program in symbolic geometric reasoning. [Lim 87b] surveys parallel processors for computer vision.

Research results are described for stereo reconstruction intended for cartography and feature analysis for areas with buildings. [Lim 87a] describes a high-level stereo correspondence system which uses surface interpretation methods to match bodies, surfaces, extended edges, and junctions. It includes an effective method to estimate shape of curved objects. [Raju 87] shows a dynamic programming solution with very low computation cost for stereo correspondence across extended edges. Problems were uncovered in the use of the Viterbi algorithm with an ordering procedure for edges from [Ohta 85]. [Nalwa 87c] demonstrates striking high resolution edge segmentation by interpolation. Lim has experimented with the high resolution technique in determining junctions for stereo.

Results are presented in inferring object shape, material, and color for object recognition in photointerpretation, target recognition, and industrial vision. We have investigated general methods of interpretation of surface shape from image boundaries under general viewpoint. [Nalwa 87a] shows that conics are images of scene edges which are conics of the same class, i.e. straight line, circle, ellipse, parabola, or hyperbola. Conics which are images of limbs (apparent edges) constrain the surface to be quadric and determine the surface up to three degrees of freedom. [Nalwa 87b] finds useful constraints from bilateral symmetry. [Binford 87] defines a generic observability model for scenes observed with sensors in the visible spectrum. Those results use boundaries of image regions; there are complementary results which use interiors of image regions. [Healey 87a] extends early results on estimating surface curvature from specularity. [Healey 87b] uses physical models to derive methods for classification of objects by material and to estimate the object's spectral reflectance, i.e. estimation with color constancy.

We have investigated color and texture description and segmentation for outdoor scenes, toward target recognition and navigation in land and air. [Vistnes 87] extends his previous work in texture with new experiments and extensive analysis for detection models. Lee and Binford have initial success in segmentation of outdoor scenes using color regions. They are now investigating segmentation of vegetation using combined color and texture. [Triendl 87] describe successful navigation of a mobile robot using stereo vision in the hallway and lobby of a building.

1 Introduction

In building a model-based vision system, SUCCESSOR, we aim to improve on limitations of ACRONYM. Components of SUCCESSOR are modeling, prediction, observation, and interpretation. An improved modeling system is operating and being extended, with these improvements: more powerful generalized cylinder primitives, set operations between primitives (ACRONYM had none); combined exact and iterative symbolic methods for display; and automated model building.

We seek these improvements in prediction: general,

*This work was supported by the Defense Advanced Research Projects Agency under contract number N00039-84-C-0211 viewpoint insensitive predictions, which were intended in ACRONYM but in reality predictions were oriented toward vertical views; incorporating new sensors and operators, i.e. depth and reflectivity, i.e. color; coarse-to-fine prediction structure and strategies; and modeling of sensors and operators.

Improvements in observation from data have been incorporated and more are underway. ACRONYM used only a subclass of ribbons from a ribbon finder which barely functioned. Improved extended edge and corner operators are used in research described here. Three dimensional results from stereo are available. Elementary but substantial use of color and texture may be near.

Interpretation in ACRONYM was combinatorial and very wasteful. Improvements are being sought: matching 3d parts instead of ribbons; incorporating coarse-fine structure and strategies; integrating multiple forms of evidence; using natural grouping into objects to restrict search.

2 System

2.1 Geometric Modeling

A generalized cylinder is a swept solid generated by sweeping a surface, the cross section, along a space curve, the spine, while transforming the cross section. [Ponce 87a] find analytic solutions for limbs (apparent edges) and cusps (terminations of limbs) for several classes of interest, a) solids of revolution; b) solids with circular cross section and curved spine, and c) straight homogeneous generalized cylinders (SHGC) (star-shaped with sweeping rule a polynomial of degree less than or equal to 5). Figure 1 shows the limbs and intersection of an SHGC from [Ponce 87b]. These are large and interesting classes of objects. These methods are potentially very fast because they are analytic. We intend to extend the classes for which limbs and cusps can be computed by extending our work in iterative methods as in [Scott 84]. A display system using ray tracing has been implemented.

The analytic equation for limbs has been used to predict invariant properties of limbs of generalized cylinders. Two algorithms have been implemented and tested on a real example for finding axes of straight homogeneous generalized cylinders. This is a beginning toward general methods for prediction and observation of parameters of generalized cylinder models.

In the modeling system, primitive parts are combined into compounds. [Ponce 87b] demonstrate calculation of intersections between straight homogeneous generalized cylinders (SHGC). They introduce a Box Tree representation for the intersection calculation for two SHGCs, and for the intersections of rays with surfaces in ray tracing. The Box Tree is like a quadtree in the parameter space of the surface; it encloses the surface in a hierarchy of enclosing boxes. Figure 2 shows a sector of a SHGC surface with a bounding box. Figure 3 shows the set difference between the two solids from Figure 1. Figure 4 shows the display made by ray tracing with a z buffer. In ACRONYM, set operations were not implemented; parts were affixed but were unaware of one another.

Another system for intersection was implemented by Chen. It was less completely implemented and had the disadvantage that its behavior was not guaranteed. In [Ponce 87b], if two surfaces intersect, their intersection will be found. In Chen's algorithm, this is not guaranteed. The surface is covered by a polyhedral web of chosen tightness of fit. Intersection of surfaces is determined by finding the intersections of edges of the polygon faces of one surface with the faces of the other.

2.2 Geometric Reasoning

[Lowry 87] describes the STRATA automatic programming system which uses parameterized theory instantiation and invariant logic to derive algorithms for formal geometric problems. These two methods are illustrated with the derivation of a variant of the Karmarker linear programming algorithm for optimization. A model of algorithm derivation is given based upon an analysis of papers in the December 1985 IU workshop proceedings.

3 Stereo Cartography

3.1 Hierarchical Stereo System

This stereo system incorporates high level interpretation of surfaces to decrease ambiguity of matching to decrease search cost and increase reliability. [Lim 87a] describes a system which builds monocular interpretation of scenes in a hierarchy of extended edge, junction, surface, and body. Surface and body interpretations follow from evidence for connectedness in space. Matching is hierarchical, i.e. bodies are matched, then surfaces within bodies, then junctions and extended edges within surfaces. The system uses extended edges from [Nalwa 84, 85]. [Lim 87] improves resolution near junctions to extract more information in areas where multiple edges interfere with the results of the edge operator. He also experimented with the high resolution version of [Nalwa 87c] which offered slight improvement at junctions. In part, performance of the stereo sys-segments without interpolation; Figure 13c shows edge segments with interpolation. Compare fine features, e.g. specularities on pins of the switch and details on the body of the switch.

For a largely bandlimited image, resolution is determined by the degree of blur. There exists a large body of work on edge-detection which recommends implicit or explicit smoothing of the image samples as the first step. The most commonly used smoothing function is the 2-D Gaussian . It can easily be shown that smoothing with this function would result in the smoothed image having an effective Gaussian blurring function with a variance equal to the sum of the variance of the original Gaussian blurring function and that of the Gaussian smoothing function. Hence, resolution between interacting step-edges, and their detection will both suffer. Also, the biases in their position and intensity estimates will increase. The extent of this deterioration will of course depend on the magnitude of σ_{smooth} (σ_{smooth} typically has a suggested lower bound of 1.0).

Although the proposed approach to detect line-edges is more general than any special purpose line-detector, we should also expect it to be more sensitive to noise with respect to false positives, particularly at high-gradient smoothly-shaded regions, false negatives, and the parameters of the true positives. Improvement in resolution is necessarily accompanied by a shrinking of the support used on the interpolated profile for detection. This results in reduced robustness to local noise-induced fluctuations in the reconstructed intensity profile. Thus, we have a trade-off between robustness and generality. This should come as no surprise considering that matched-filtering, wherein noisy patterns are categorized based on their closest "match" to noiseless representatives of the different classes, is well known to be optimal.

5 Object Recognition

5.1 Line-Drawing Interpretation: Straight Lines and Conics

Line-drawings are representations of edges in an image. An edge in an image may be caused by various events in the scene. The scene-event may be a surface-tangent discontinuity, a depth discontinuity, a surface-reflectance discontinuity or an illumination discontinuity (shadow). Notice that a depth discontinuity may also simultaneously be a surface-tangent discontinuity. We distinguish this from a continuous-surface-tangent depth discontinuity. While surface-tangent discontinuities, illumination discontinuities, and surface-reflectance discontinuities constitute viewpoint-independent scene-edges, continuous-surface-tangent depth discontinuities are viewpoint-dependent.

Line-drawings corresponding to orthographically projected images of man-made scenes often contain instances of straight-lines and conic-sections, i.e., ellipses, parabolas, and hyperbolas. We investigate constraints imposed on the scene by such instances under the assumption of *general* viewpoint, i.e., under the assumption that the mapping of the viewed surface onto the line-drawing is stable under perturbation of the viewpoint within some open set on the Gaussian sphere.

It is assumed that the projected surfaces are piecewise C3, i.e., the surfaces comprise finitely many C3 patches, each bounded by a finite piecewise C3 curve. This restriction does not significantly constrain the domain as long as we do not require a priori knowledge of the C3 surface-patch boundaries. The other assumptions are explicitly stated in the theorems. The straight-lines and conic-sections in the line-drawing need not be continuous for the theorems to be valid, i.e., the constraints imposed on the viewed surface hold even if the curves in the line-drawing are fragmented.

Theorem 1 : Straight-lines in a line-drawing obtained under orthographic projection with a general viewpoint are projections of straight-lines in space.

Theorem 2 : Ellipses, parabolas and hyperbolas in a line-drawing obtained under orthographic projection with a general viewpoint are projections of ellipses, parabolas and hyperbolas, respectively, in space.

Theorem 3 : Circles in a line-drawing obtained under orthographic projection with a general viewpoint are projections of circles in space which are confined to lie in planes parallel to the image plane.

Theorem 4 : Scene events which orthographically project, under general viewpoint, onto straight-lines or conic-sections in line-drawings must either be completely viewpoint-independent or completely viewpoint-dependent.

Theorem 5 : Continuous-surface-tangent depth discontinuities which orthographically project, under general viewpoint, onto straight-lines in line-drawings can be described locally by developable surfaces.

Theorem 6 : Continuous-surface-tangent depth discontinuities which orthographically project, under general viewpoint, onto conic-sections in line-drawings can be described locally by quadric surfaces. Specifically, continuous-surface-tangent depth discontinuities projecting onto ellipses by ellipsoids or hyperboloids of one sheet, onto parabolas by elliptic paraboloids or hyperbolic paraboloids, and onto hyperbolas by hyperboloids of one sheet or hyperboloids of two sheets. Furtem is linked to improved edge segmentation, extended edges, and junctions which the algorithm uses. Figure 5 shows the original stereo pair for a scene containing curved surfaces and blocks. Figure 6 shows extended curves extrapolated to junctions. Figure 7 shows segmented surfaces; there are only 11 surfaces and 7 bodies. Figure 8 shows a projection of depth data from stereo matching.

Because there are few bodies with rather strong distinctions, matching has very low ambiguity. The long term plan is to extend incorporation of surface interpretation techniques, especially quasi-invariants in matching.

The system incorporates an elegant analysis and implementation of estimation of the 3d shape of curved surfaces from their limbs in stereo images. It is obvious tht two views of apparent edges correspond to different curves on the surface. One ad hoc approach is to take the nearest intersection of rays. In reality, the two limbs are rather far apart on the surface for large baseline stereo. A more accurate procedure is to incorporate the natural constraints, that two views of two limbs within an epipolar plane give four rays along which the surface is tangent. The resulting surface must be tangent to these four rays. The most general curve in the epipolar plane which is determined is a conic. A solution has been implemented. Figure 9 shows conics constrained to fit four tangents.

3.2 Stereo using dynamic programming

[Raju 87] describe a study of dynamic programming using the Viterbi algorithm. The principal result is that computation is drastically reduced over [Ohta 85] which was a rather direct extension of the Viterbi algorithm across epipolar planes. A serious problem of missing matches was unearthed which may be quite fundamental.

The Viterbi algorithm cannot tolerate order reversal between matched sequences, i.e. surfaces in left and right views. Edges and surfaces in the plane are not ordered. [Ohta 85] describe an ordering scheme. It appears that the Viterbi algorithm is incompatible with this ordering scheme. That is, order reversals occur which mean that necessary matches are not considered. The problem becomes very serious in complex scenes, resulting in about 25successful matches in the complex Washington stereo scene. Figure 10 shows matched edge segments for the stereo pair in Figure 6 above. Note that two edges of the pentagonal prism are not matched because of this problem.

The evaluation criterion for matching surfaces is the product of similarity functions incorporating quasi-invariants for surface orientation (interval) and edge orientation, along with edge contrast. The evaluation function and search method are related to our earlier stereo implementations [Arnold 80, 82; Baker 81].

4 Edge-Detector Resolution Improvement by Image-Interpolation

[Nalwa 84] seeks to demonstrate how so-called line-edges may be detected by a step-edge detector. This is achieved by reconstructing the underlying image-intensity surface. Line-edges need to be discovered nevertheless, not because it is not possible to detect them as a pair of step-edges, but because we want to accurately recover the parameters of the underlying steps. As is evident, the interaction between the images of the component step-edges will result in systematic errors in their position and intensity estimates — the discovery of line-edges is needed to remove these biases. (In general, non-zero slopes on the two sides of step-edges will also bias the parameter estimates.) We concern ourselves here with the detection aspect and do not address in any detail the issues pertaining to the systematic errors.

In [Nalwa 84] it was illustrated that 3 symmetric samples were insufficient to detect a 1-D step-edge owing to inherent ambiguities. It is clear, however, that if we had available the continuous signal itself, then our window size would not be thus restricted, i.e. a 3 pixel window on the continuous signal has no such ambiguity. But, if our signal is by and large bandlimited, then we can use an interpolation filter to reconstruct the underlying continuous signal. This is a reasonable assumption if σ_{blur} , for the effective Gaussian blurring function associated with the imaging system, is no less than 0.5 pixel. Hence, although we do require more than 3 samples to detect a step-edge, it is not necessary that we be able to isolate the step-edge within such a window. We can instead use the samples to reconstruct the continuous signal and then detect step-edges using smaller windows.

. All the foregoing arguments carry over to 2-D. In practice, we need not reconstruct the continuous signal. Interpolation onto a finer grid often suffices. For purposes of demonstration, we chose our new grid to have the half the original inter-pixel spacing. Figure 12 shows detection and estimation of step edges separated by 3 pixels, 2 pixels, and 1 pixel. Figure 13a shows the original picture of a bin of parts; Figure 13b shows edge thermore, these quadric surfaces are completely determined, up to three degrees of freedom, by the conic-sections in the line-drawings.

Theorem 7 : Depth discontinuities which orthographically project, under general viewpoint, onto circles in line-drawings can be described locally by spheres whose radii are identical to the radii of the circles in the line-drawings.

5.2 Line-Drawing Interpretation: Bilateral Symmetry

Symmetry manifests itself in numerous forms throughout nature and the man-made world. It is frequently encountered in line-drawings corresponding to edges in images. We seek to discover constraints on the imaged surfaces from instances of bilateral (reflective) symmetry in line-drawings obtained from orthographically projected images. A *general* viewpoint is assumed, i.e., it is assumed that the mapping of the viewed surface onto the line-drawing is stable under perturbation of the viewpoint within some open set on the Gaussian sphere. It is shown that the only planar figures which exhibit bilateral symmetry under orthographic projection with a *general* viewpoint are ellipses and straight-line segments.

We show that the orthographic projection of a surface of revolution exhibits bilateral symmetry about the projection of the axis of revolution, irrespective of the viewing direction. Further, it is shown that whenever the back-projection of the symmetry axis is invariant in space under perturbation of the viewpoint, the bilaterally symmetric line-drawing is the orthographic projection of a local surface of revolution. The axis of revolution is in the back-projection of the symmetry axis. Various line-drawing configurations for which the back-projection is invariant are detailed.

It is conjectured that isolated ellipses and isolated straight-line segments are the only bilaterally symmetric line-drawings whose back-projections of the symmetry axes are not invariant under perturbation of the viewpoint. Throughout, only surface regions local to the scene-events corresponding to the lines are constrained.

Now we detail various line-drawing configurations for which the back-projection of the symmetry axis is invariant under perturbation of the viewpoint. Case I. Consider a bilaterally symmetric line-drawing which has two parallel straight-line segments, one on either side of the symmetry axis. For stable bilateral symmetry the two straight scene-edges must be viewpoint-dependent and further must locally lie on a single right-circular cylinder. The back-projection of the symmetry axis is the axis of this cylinder. Case II. Consider a bilaterally symmetric line-drawing which has two non-parallel straight-line segments, one on either side of the symmetry axis. For stable bilateral symmetry the two straight scene-edges must be viewpoint-dependent and further must locally lie on a single right-circular cone. The back-projection of the symmetry axis is the axis of this cone. Case III. Consider a bilaterally symmetric line-drawing which has an elliptical segment (may be fragmented) bisected by the symmetry axis. It is easy to see that the back-projection of the centroid of the (completed) ellipse must continue to lie on the back-projection of the symmetry axis under perturbation of the viewpoint. Similarly, a tangent-discontinuity on the symmetry axis in a line-drawing must be the projection of a conical tip in space which continues to lie on the back-projection of the symmetry axis under perturbation of the viewpoint. As any two points in space determine a line, two independent events from among ellipses bisected by the symmetry axis and tangent-discontinuities on the symmetry axis fix the back-projection of the symmetry axis. Further investigation would almost certainly lead to more relaxed conditions for the invariance of the back-projection of the symmetry axis. [Nalwa 87b] provides several examples of bilaterally symmetric line-drawings for which we can presently deduce local surfaces of revolution.

It is our conjecture that isolated ellipses and isolated straight-line segments are the only bilaterally symmetric line-drawings whose back-projections of the symmetry axes are not invariant under perturbation of the viewpoint. What we do know is that ellipses and straight-line segments are the only planar figures which exhibit bilateral symmetry under orthographic projection with a *general* viewpoint (see Appendix). Ellipses project onto ellipses and straight-line segments onto straight-line segments. It follows that all viewpoint-independent scene-edges which are isolated ellipses or isolated straight-line segments exhibit bilateral symmetry under projection. These, of course, are not local surfaces of revolution. The viewpoint-dependent scene-edges which project onto isolated ellipses are either local ellipsoids or local hyperboloids of one sheet [Nalwa 87b]. These, in general, are also not local surfaces of revolution. From the result in [Koenderink 82], that terminations of occluding contours are always concave, it follows that no viewpoint-dependent scene-edge may project onto an isolated straight-line segment.

It might interest the reader to learn that this work was prompted by the observation in [Nalwa, 1987] that surfaces which orthographically project, under *general* viewpoint, onto circles in line-drawings can be described locally by spheres.

Finally, we note that we have made no effort whatsoever to suggest schemes for the detection in line-drawings of bilateral symmetry, in general, and ellipses and straight-lines, in particular. Neither have we analyzed the robustness of the conclusions which may be drawn from bilateral symmetry in line-drawings, i.e., whether approximate symmetry translates to approximate local surfaces of revolution. Practical considerations demand that these issues be addressed.

5.3 Surface Materials and Color

An important intrinsic property for object identification is surface material and reflectance. [Healey 87b] use physical models for reflection of light from colorant layers. Figure 14 shows surface and body scattering for an inhomogeneous material. The problem of optimal sensor spectral response is addrssed. Homogeneous materials, e.g. metals, have reflection only from the surface. Inhomogeneous materials, e.g. plastics, have surface reflection and reflection from colorant particles in the medium. Using the previous analysis of specularities [Healey 87a], if a color discontinuity is found at specularity, the material is probably inhomogeneous, otherwise homogeneous. If homogeneous, the color distinguishes gold and silver for example. A method is proposed for estimating surface spectral reflectances. Classification and estimation of reflectance were both implemented. Experimental results confirm these estimates. Figure 15 a,b,c show Fresnel reflection, diffuse reflection, and computed reflectance for a blue cup.

5.4 Local shape from specularity

[Healey 87a] analyze specularities using the Torrance-Sparrow model. They derive predictions to be used in a model-based system. More important, they determine programs to extract specularities and from them to estimate local curvatures of the surface at the specularity. Figure 16 shows the specular intensity surface for a curved surface. Careful measurements demonstrate detailed agreement in estimations of curvature. Table 1 and 2 show actual and measured surface statistics estimated from specularity. Agreement is within 2.5%.

6 Navigation

6.1 Mobile Robot Navigation

[Triendl 87] describe successful navigation by the Stanford Mobile Robot using stereo vision and motion. The robot went down a hallway, toured the lobby, and returned down the hallway, a distance of 35 meters, which it traveled in one meter steps at about 20 seconds per step plus motion time. Only one second of this time was used by vision, the rest by planning and system inefficiencies. The vision system has a weak model of generic buildings, including a horizontal floor, with vertical walls and doors. Walls intersect walls. In navigating in the lobby, it overcame several complexities including an isolated pillar which causes violations of ordering constraints in stereo, and large windows.

Figure 17 shows a view of the hallway as seen by stereo cameras of the mobile robot. In the model, edges are vertical or horizontal. The vision system looks only for vertical edges along a narrow horizontal band. It limits its attention in order to get reasonable computation time. Vision computation is done offboard on a VAX in FranzLisp; the vision system is being moved to a Symbolics Lisp machine. From two views of the few vertical edges together with properties of surfaces between, it attempts to instantiate the model of a hallway with walls and doors. It integrates sequences of stereo pairs to register edges between views and to determine its actual motion between stereo pairs. Figure 18 shows a birdseye view of an instance of the hallway model from a single view. The system calculated free space and planned very simple moves. Figure 19 shows commanded motions compared with visual estimates of actual motions for a sequence of moves down the hallway. Estimates of locations of features in multiple views were combined to reduce accumulation of errors.

7 Texture

[Vistnes 87a] extends the work reported in [Vistnes 85] on human detection of dotted lines and curves embedded in random-dot surrounds. In the previous experiments, the random surrounds were made uniform to avoid spurious dot clusters that might confuse subjects looking for dotted lines. However, this approach made analysis difficult. The experiments were repeated using a purely random dotted surround, and the analysis extended.

[Vistnes 87a] measures how much point scatter (orthogonal to the axis of the dotted line) could be tolerated, and how much curvature in an arc could be tolerated. The main factor that determines our ability to detect such patterns is the relative separation of target dots compared to surround dots, but there is a significant effect of density: we can detect curves with more jaggedness and curvature in dense backgrounds than in sparse.

He presents a statistical model for line and curve detection, and uses the model to make qualitative predictions about human performance in this task. The model is based on estimating the density of dots surrounding an elongated center region, counting the number of dots in the center, and calculating the probability that those dots fell in the center region at random. When this probability is low, we infer that the dots in the center region have a non-accidental origin, i.e., correspond to some structure in the scene. Predictions based on this model agree, within their limits, with empirical results. Predictions for curve detection, based on detection of lines, agree with data on human curve detection, supporting a hypothesis that curves are detected by line detectors.

Computer simulations, in which line-detection operators of the type proposed here are applied to a line detection task, are discussed. The results compare well to human results and to predictions.

8 References

[Arnold 80] R.D.Arnold, T.O.Binford; "Geometric Constraints in Stereo Vision" in Image Processing for missile guidance; Proc SPIE, 238, 1980, 281-292.

[Arnold 82] R.D.Arnold; "Automated Stereo Perception"; Tech report AIM-347, Stanford University,1982.

[Baker 81] H.H.Baker; "Depth from edge and intensity-based stereo"; Proc 7th IJCAI, 1981.

[Healey 87a] G.Healey, T.O. Binford; "Local shape from specularity"; Proceedings Image understanding workshop, Feb, 1987.

[Healey 87b] G.Healey, T.O. Binford; "The role and use of color in a general vision system"; Proceedings Image understanding workshop, Feb, 1987.

[Koenderink 82] J.J.Koenderink, A.J.van Doorn; "The shape of smooth objects and the way contours end"; Perception, V 11, 1982, 129-137.

[Lim 87a] H.S.Lim, T.O.Binford; "Stereo Correspondence; A Hierarchical Approach"; Proceedings Image understanding workshop, Feb, 1987.

[Lim 87b] H.S.Lim, T.O.Binford; "Survey of Parallel Computers"; Proceedings Image understanding workshop, Feb, 1987.

[Lowry 87] Michael Lowry; "Automatic Programming for IU Applications"; Proceedings Image understanding workshop, Feb, 1987.

[Nalwa 84] V.Nalwa; "On Detecting Edges"; Proceedings of the Image Understanding Workshop, 1984.

[Nalwa 85] V. S. Nalwa, E. Pauchon; "Algorithms for edgel-aggregation and edge-description"; Proceeding: Image understanding workshop, December, 1985.

[Nalwa 87a] V.Nalwa; "Line-Drawing Interpretation: Straight Lines and Conics"; Proceedings Image understanding workshop, Feb, 1987.

[Nalwa 87b] V.Nalwa; "Line-Drawing Interpretation: Bilateral Symmetry"; Proceedings Image understanding workshop, Feb, 1987.

[Ohta 85] Y. Ohta, T. Kanade; "Stereo by Intra-and-Inter-scanline search using dynamic programming"; IEEE Trans., PAMI, March 1985.

[Ponce 87a] J.Ponce, D. Chelberg, W.Mann; "Analytical properties of generalized cylinders and their projections" Proceedings Image understanding workshop, Feb, 1987.

[Ponce 87a] J.Ponce, D. Chelberg; "Localized intersections computation for solid modelling with straight homogeneous generalized cylinders". Proceedings Image understanding workshop, Feb, 1987.

[Raju 87] G.V.S.Raju, T.O.Binford, S.Shekar; "Stereo Matching using Viterbi Algorithm"; Proceedings Image understanding workshop, Feb, 1987.

[Scott 84] R. Scott; "An Algorithm to Display Generalized Cylinders"; Proceedings of the Image Understanding Workshop, May 1983.

[Triendl 87] E.Triendl, D.J. Kriegman; "Vision and Visual Exploration for the Stanford Mobile Robot"; Proceedings of the Image Understanding Workshop, Feb 1987.

[Vistnes 85] R. Vistnes; "Detecting structure in random-dot patterns"; Proceedings Image understanding workshop, Feb, 1987.

[Vistnes 87a] R. Vistnes; "Detecting dotted lines and curves in random-dot patterns"; Proceedings Image understanding workshop, Feb, 1987.

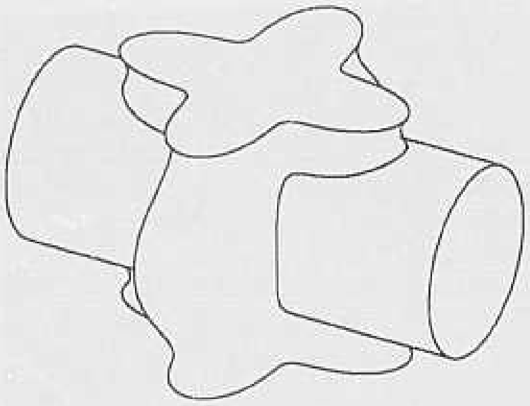


Fig. 1. Union of cylinder and SHGC

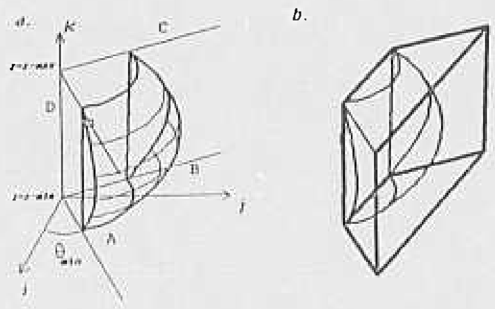


Fig 2. Sector of SHGC and enclosing box

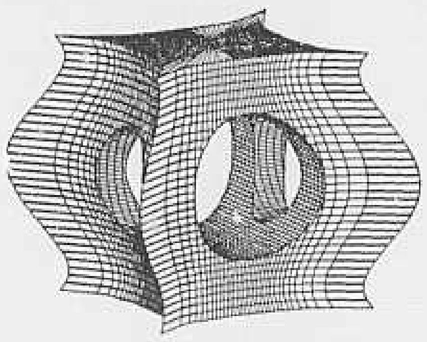


Fig 3. Difference of SHGC and cylinder.

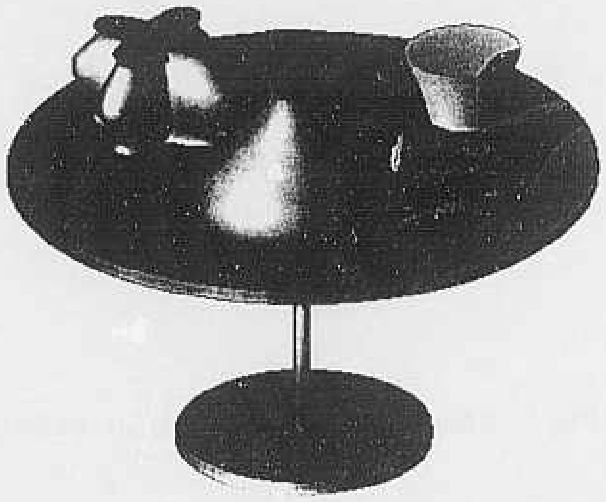


Fig 4. Ray traced image of SHGCs.

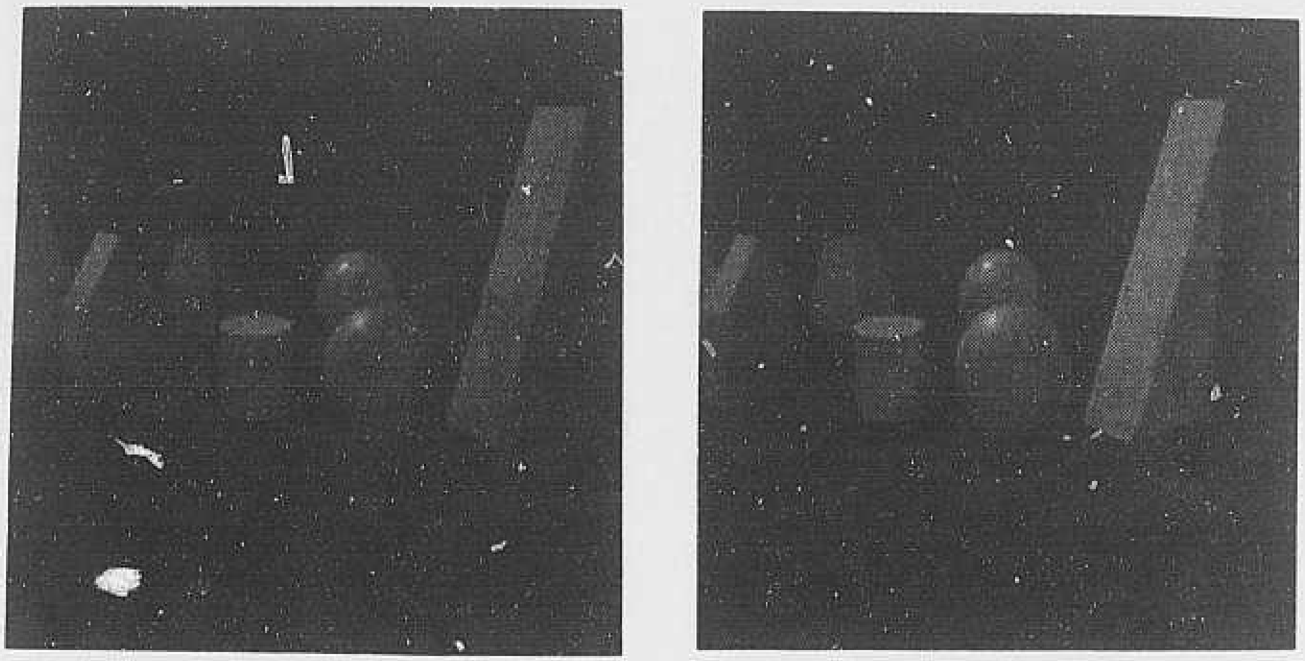


Fig 5. stereo pair of curved objects and blocks.

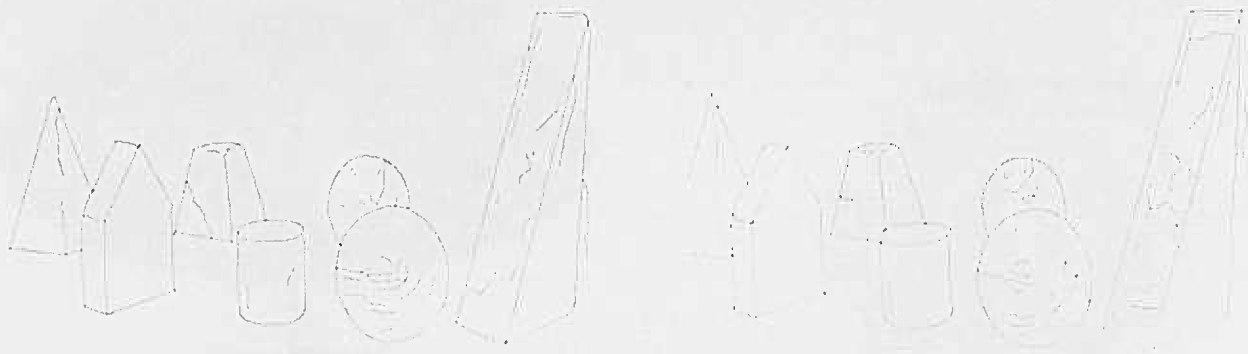


Fig 6. edges extended to junctions; left and right.

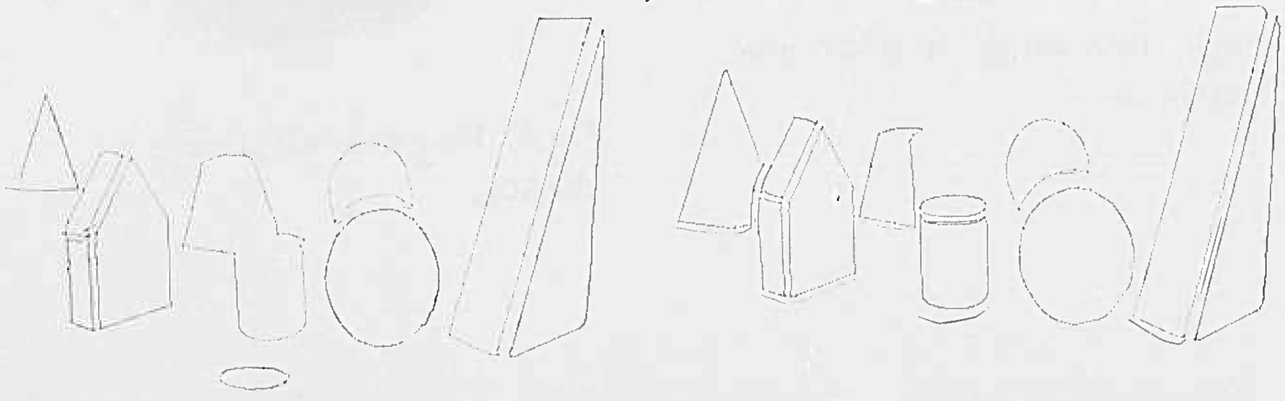


Fig 7 surfaces left and right.

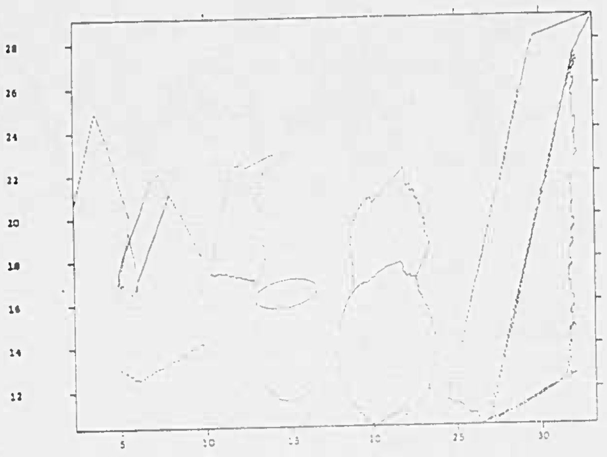


Figure 8. Projection of Depth Map from Fig 5.

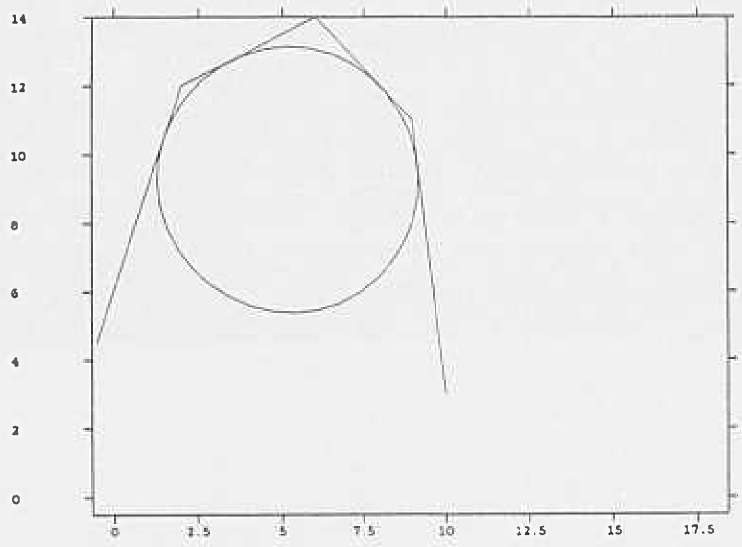


Figure 9a. Fitting Conics to Four Tangents.

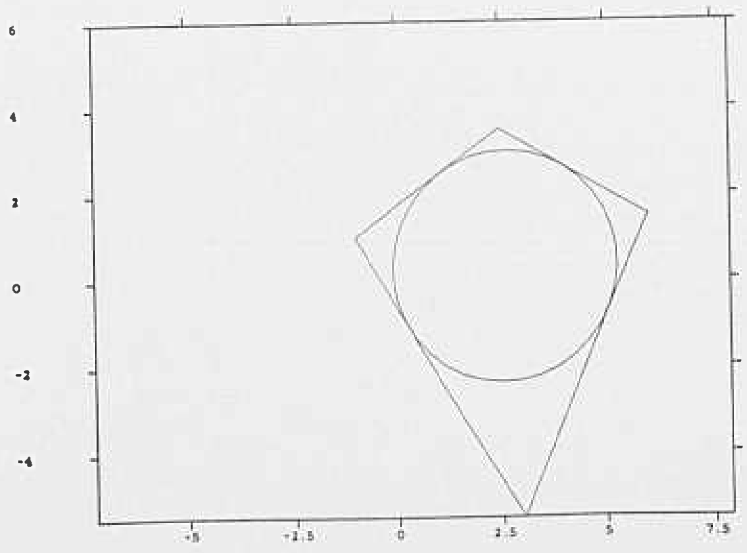


Figure 9b. Fitting Conics to Four Tangents.

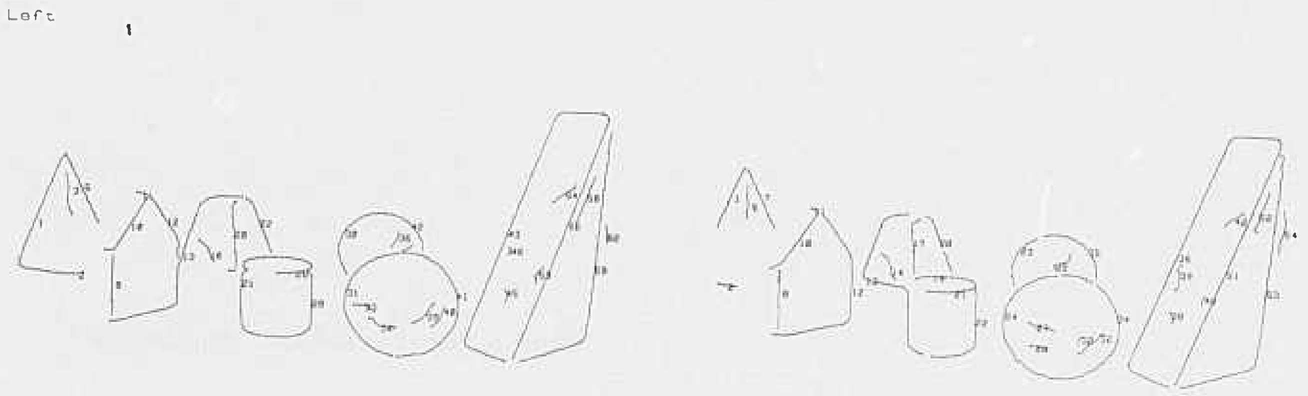


Figure 10. Matching surfaces from dynamic programming.

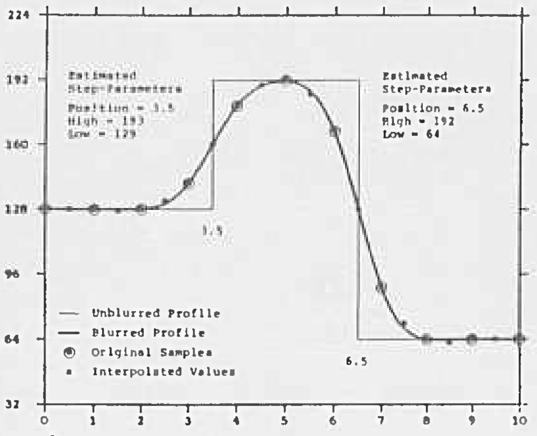


Fig. 12a Detection and estimation of step-edges spaced 3 pixels apart (σ_{blur} = 0.6).

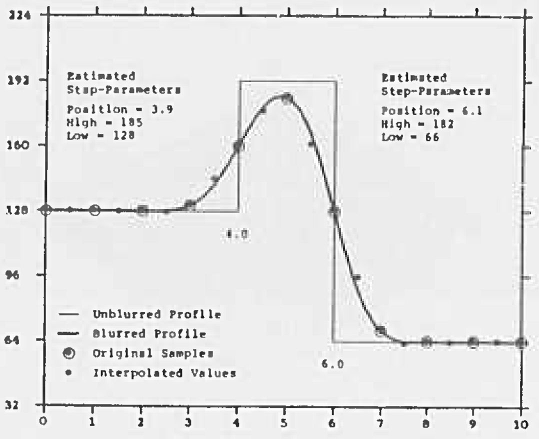


Fig. 12b Detection and estimation of step-edges spaced 2 pixels apart (σ_{blur} = 0.6).

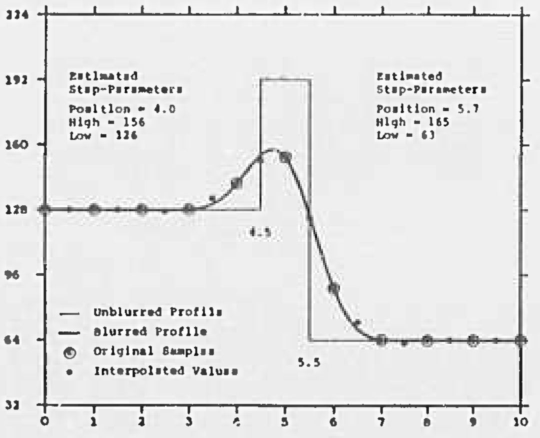


Fig. 12c Detection and estimation of step-edges spaced 1 pixel apart (σ_{blur} = 0.6).

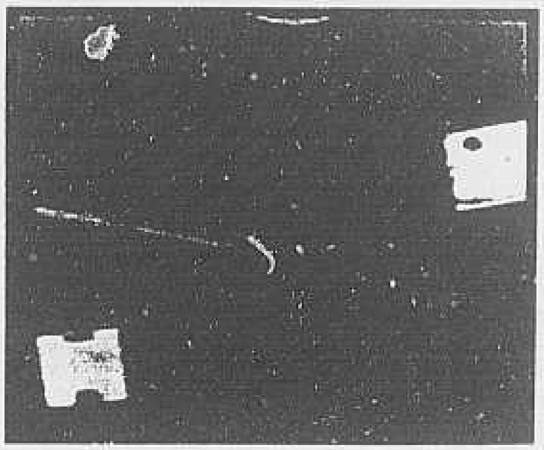


Fig. 13a Bin of Parts : Original Image (128 x 128)

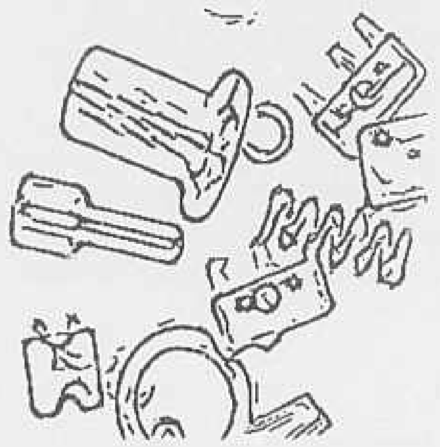


Fig. 13b Bin of Parts : Edge Image without Interpolation (adapted from [9])

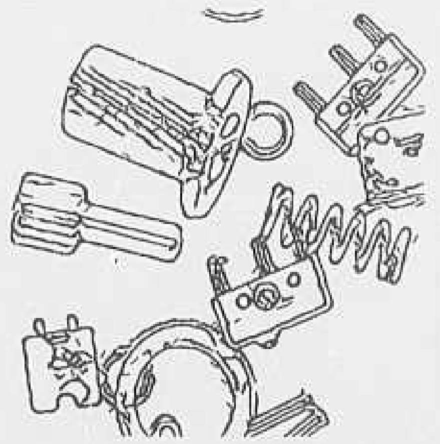


Fig. 13c Bin of Parts : Edge Image with Interpolation

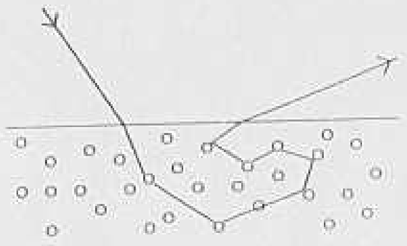


Figure 14 Scattering by Colorant Particles

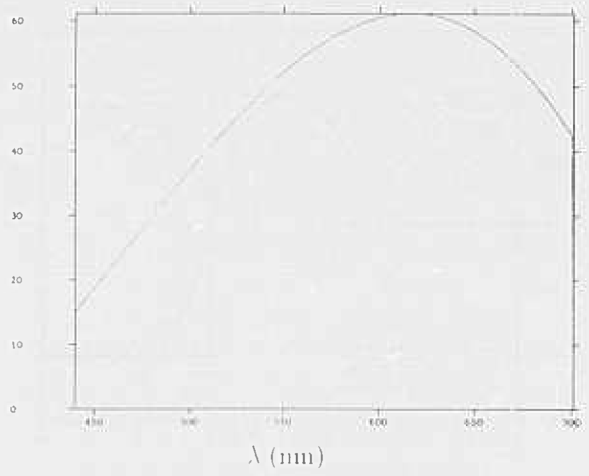


Figure 15c Fresnel Reflection from blue cup

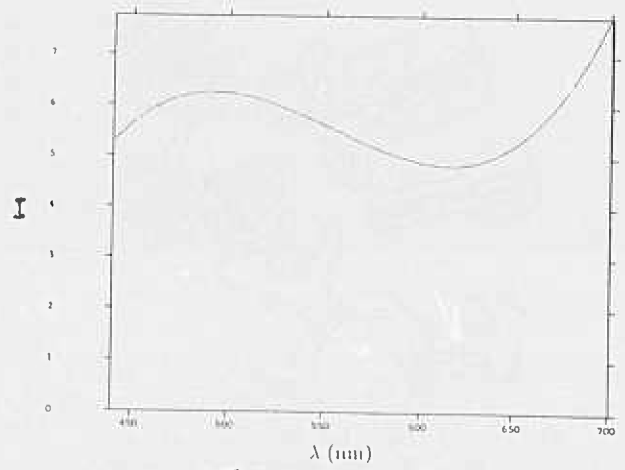


Figure 15 Diffuse Reflection from blue cup

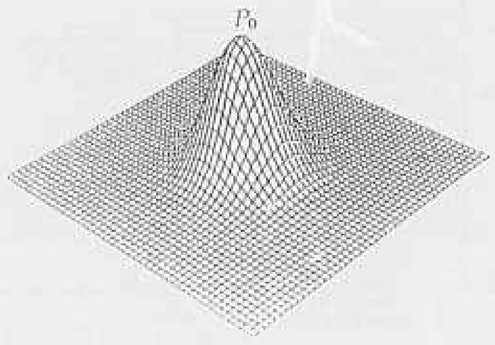


Figure 16. Specular Intensity Surface.

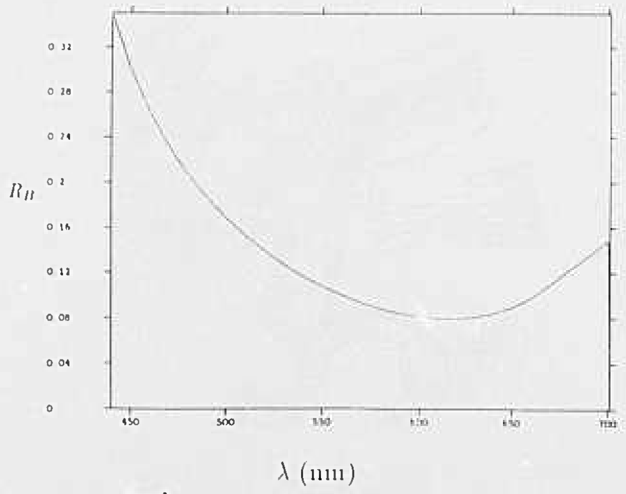


Figure 15b Computed Reflectance for blue cup

Object	κ_1	θ_1	κ_2	θ_2
cylinder$_1$	0.286	0.0	0.0	1.571
cylinder$_2$	0.400	0.0	0.0	1.571
cylinder$_3$	1.333	0.0	0.0	1.571
sphere$_1$	0.500	—	0.500	—

Table 1. Actual Surface Statistics

Object	κ_1	θ_1	κ_2	θ_2	Error
cylinder$_1$	0.297	0.0	0.001	1.571	3.9%
cylinder$_2$	0.397	0.0	0.001	1.571	0.9%
cylinder$_3$	1.356	0.0	0.002	1.571	1.7%
sphere$_1$	0.514	0.0	0.534	1.571	2.8%

Table 2. Computed Surface Statistics

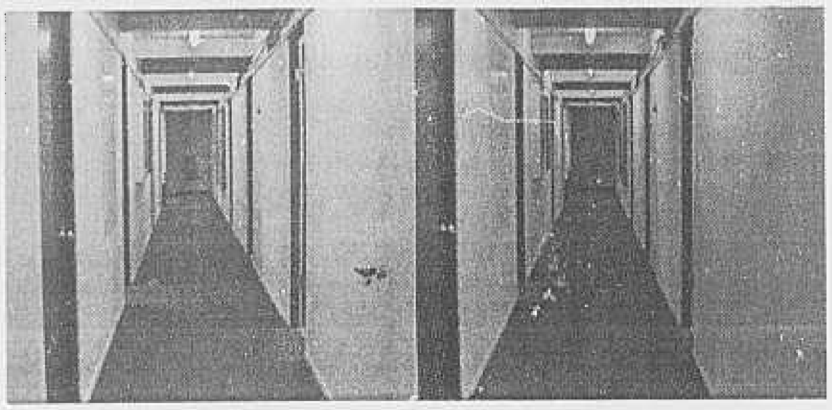


Figure 17. View of the Hallway as seen by Stereo Cameras.

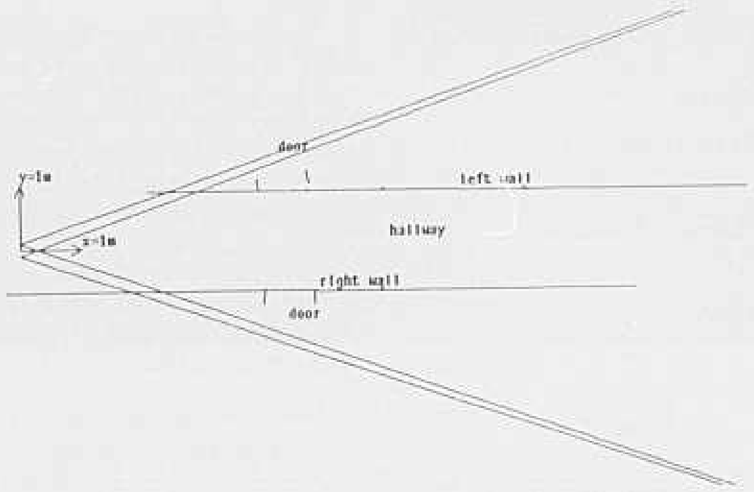


Figure 18 *. Wall and doors from figure 2*

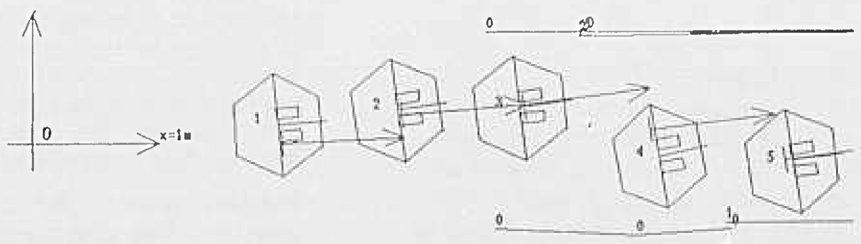


Figure 19 *Stereo results and navigation by vision. Robot icons show the locations derived from vision while arrows show the incremental motions predicted from revolutions of the vehicles wheels. Figure 2 showed the image taken from the first position. Stereo results from this composite are labeled 0.*

Image Understanding Research at CMU

Takeo Kanade

Computer Science Department
Carnegie-Mellon University
Pittsburgh PA 15213

Abstract

The CMU Image Understanding Program covers a wide range of topics ranging from the basic issues in calibrated imaging, color, and shape to the system issues in developing demonstrable vision systems. This report reviews our progress since the December 1985 Image Understanding Workshop in the following areas:

- *Vision and System for Navigation*
- *Model-Based Vision and AI Techniques*
- *Color Understanding*
- *Representing Uncertainty in Low-Level Vision*
- *Parallel Vision on Warp*
- *Calibrated Imaging Laboratory*

1. Vision and System for Navigation

In the Strategic Computing Vision program, Carnegie Mellon has been working on vision techniques for navigation and their integration into a demonstrable vehicle [8, 9]. During the last year, we have taken substantial strides by constructing the Navlab vehicle, implementing the CODGER software system for integration, developing the CMU navigation architecture and some of sensory information processing modules, and finally demonstrating the Sidewalk and the Park navigation runs.

1.1 Navlab: Navigation Laboratory

The Navlab, short for Navigation Laboratory, is a self-contained laboratory for navigational vision system research. We had been using the Terregator, which is a six-wheeled vehicle teleoperated from a host computer through a radio link. The Terregator had been a reliable workhorse for small-scale experiments, such as the Campus Sidewalk navigation system [3]. However, we have outgrown its capabilities. As we began to experiment with sensor fusion, the Terregator ran out of space and power for multiple sensors. When we wanted to expand our test areas, communications to a remote computer in the lab became more difficult. And as the experiments became more sophisticated, we found it more productive for the experimenters to test or debug new programs near or in the vehicle, instead in a remotely located laboratory. All these factors culminated in the design and construction of the Navlab.

Navlab is based on a commercial van chassis, with hydraulic drive and electric steering. Computers can steer and drive the van by electric and hydraulic servos, or a human driver can take over control if necessary. It is even licensed in Pennsylvania, so we can drive it by hand to our remote test sites. Once there, the driver pulls a switch and an onboard microprocessor assumes control of the steering and drive. The Navlab has room for researchers and computers on board, and has enough power and space for all our existing and planned sensors. This gets the researchers close to the experiments, and eliminates the need for video and digital communications with remote computers.

Vital features and statistics of the Navlab include:

- **Onboard computers**: In addition to a micro processor system for low-level control of vehicle motion, the Navlab includes currently three SUN 3 workstations. A unit of the Warp processor is planned to be installed in early 1987.
- **Onboard sensors**: Currently, above the cab we have a color camera and an ERIM laser range finder. In the future we will augment the sensors with another color camera, a pan/tilt mount for cameras, a separate pan mount for ERIM, and an inertial navigation sensor.
- **Onboard researchers**: There will always be a safety driver in the driver's seat. There is room for a terminal in front of the passenger's seat. The main work area, in the back, includes a full-length desk (opposite the equipment racks), with overhead shelf for monitors.
- **Onboard power**: We carry two 5500 watt generators, plus power conditioning and battery backup for critical components.

Navlab is much more controllable than the Terregator. The first computer controller for the Navlab is relatively primitive, but is enough to let our sensor interpretation work make its first tests. The controller will soon evolve to accept queues of commands, to interface to an inertial positioning unit and perhaps to a GPS satellite positioning system, and to watch vital signs such as computer temperature and vehicle hydraulic pressure.

1.2 CODGER System

We have implemented the CODGER system (COmmunications Database with GEometric Reasoning) [13]. The CODGER system consists of a central database (*Local Map*), a process that manages this database (*Local Map Builder*), and a library of functions for accessing the data. The CODGER system provides software support for interfacing the various perception, planning, and control modules of the vehicle into a single autonomous system. More specifically, CODGER provides:

- *Parallelism:* CODGER enables modules to run in parallel on a cluster of SUN-3's interconnected with an EtherNet. The modules synchronize themselves through CODGER's central database.
- *Data Representation:* CODGER provides a flexible token-based data representation scheme as well as a number of geometric functions for indexing this data.
- *Sensor Fusion:* CODGER provides a mechanism for fusing geometric data from a number of sensors.

1.3 Navlab Architecture and Sensory Information Processing

Built on top of the CODGER system is the Navlab Module Architecture, whose major information flow is illustrated in Figure 1. Each box in the diagram corresponds to one or more modules in the system. Lines indicate information flow between the modules through the CODGER database. The CAPTAIN parses a human-specified mission (point to point) and oversees its execution. Mission steps are passed to the MAP NAVIGATOR, which searches a map database to find the best route satisfying the mission step. The resultant route is decomposed into *route segments*, each corresponding to a uniform mode of driving (e.g., road following, turning in an intersection, cross-country, etc.). Route segments are passed to the PILOT, which oversees their execution. The PILOT decomposes each route segment into a number of perceivable pieces called *driving units*. On each driving unit, the PILOT performs four operations. First, it calls upon the PERCEPTION system to recognize it. Second, it calls upon the PERCEPTION system (range finder) to scan it for obstacles. Third, it calls upon the local path planner to plan a path through it. Finally, it calls upon the HELM to drive the vehicle through it. The four PILOT operations form a *pipeline*; during normal driving the PILOT works on four driving units at a time. The paper "The CMU Navigational Architecture" (Stentz and Goto) [14] in this volume describes more detail.

Currently we use color vision and range imagery from the ERIM sensor [17, 5]. In color vision, color and texture features of road and non-road regions are adapted continuously as the vehicle moves. We locate the road region by means of a robust Hough transform which uses two parameters to represent the local road geometry relative to the vehicle. The ERIM range imagery is used to locate smooth navigable regions and three-dimensional obstacles on and off the road. The results of these sensory information processing modules are sent to the *Local Map* to be fused and used by other modules. Recent progress in the sensory information processing is summarized in the paper "Vision and Navigation for the Carnegie Mellon Navlab" (Thorpe, Shafer and Kanade) [16] in this volume.

1.4 Scenarios and Capabilities

We have tested our system at two sites, the Carnegie-Mellon University campus and an adjoining park, Schenley Park. The CMU campus test site has a sidewalk network including intersections, stairs and bicycle slopes. The Schenley Park site has curved narrow roads in an area well populated with trees. We have used two experimental vehicles, the Navlab in Schenley Park and the Terregator on the CMU campus. Both vehicles are equipped with a color TV camera and an ERIM laser rangefinder. The Navlab carries three SUN-3 general purpose computers on-board. The Terregator is linked to SUN-3s in the laboratory with video and radio communications.

Currently, the system has the following capabilities:

- Execute a prespecified user mission over a mapped network of sidewalks, including turning at the intersections and driving up the bicycle slope.
- Recognize simple objects, such as the stairs and intersections.

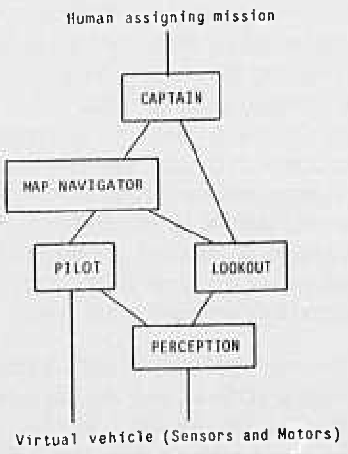


Figure 1: Navlab Block Diagram

- Drive on unmapped, curved, ill-defined roads in the park under fairly varied conditions including strong shadows, wet surface and leaves.
- Detect navigable regions and 3D obstacles from range images
- Stop when obstacles are found on the road, resume the motion when they are removed. Avoid obstacles if possible.
- Drive continuously at up to 200mm/sec.

The papers [16, 14] included in the Proceedings discusses the current state, the problems and the future plan in more detail.

2. Model-Based Vision

In the area of model-based vision, we have been pursuing automatic acquisition of object recognition algorithms and a framework for representation and control for knowledge-based vision systems. In this type of research, we are also exploring application of object oriented programming techniques to vision. The object-oriented programming allows us to organize the procedures which affect each object with the definition of the object and to have the necessary procedures automatically executed when an object is created or modified. This simplifies separation of general knowledge, such as geometric models, from the data specific to a particular run of the program.

2.1 Generation of 3D Object Recognition Algorithms from Models

Ikeuchi has been working on a method to generate 3D-object recognition algorithms from a geometrical model. His application task is bin picking, ie. recognizing an instance of the object in a jumble and picking it by a robot. His method consists of compile-time processing and run-time processing.

Given a 3D solid model of an object, the compile-time processing first generates apparent shapes of the object under various viewer directions. Those apparent shapes are then grouped into distinctive aspects (ie., different apparent shapes the object can take) based on dominant visible faces and other features. This grouping then is used to generate recognition algorithms to be executed at the run time. The algorithms are represented in the form of a two-part interpretation tree: the first part represents how to classify a target region in an image into one of the aspects, and the second part represents how to determine the precise attitude of the object within that aspect. At the run time, images of the scene are processed to compute features, and each part of the scene is interpreted by following the process specified by the interpretation tree.

This method has been applied to objects which include both planar and cylindrical surfaces, and the bin picking has been demonstrated [7]. As sensory data, we have used surface orientations from photometric stereo, depth from binocular stereo using oriented-region matching, and edges from an intensity image. Features used in interpretation include inertia of a region, relationship to the neighboring regions, position and orientation of edges, and extended Gaussian images. The system not only locates instances of the object, but does it also creates a three-dimensional scene description that other processes, such as grasping motion generator, can use to perform necessary geometrical reasoning.

Historically, and even today, many computer vision programs are "hand" written by a vision programmer who, given an example object, thinks hard and comes up with useful features, an effective order in which to use them, and reliable decision criteria. When written correctly, the resulting programs are very effective and efficient because they take considerable prior knowledge of the object into account. In this sense, they constitute model-based vision. Naturally, however, it takes a long time for a capable programmer to write such good special programs. An alternative approach is to develop a general model-based vision program which takes a model of the object and recognizes the scene by reasoning about various properties and relationships based on the model. Though general, such a system pays the cost of generality; that is, it is less efficient and less effective than specially tailored programs because of the difficulty in taking advantage of special cases which are only true with the particular object. Ikeuchi's approach provides a third alternative: develop a "general" program which takes a model of an object and generates (compiles) a "special" run-time program tailored for the object.

This approach will also have an impact on automatic learning in vision. During the course of the research, Ikeuchi has developed a set of rules (mostly heuristic) which guide the decisions about what features are to be used in what order to generate an efficient and reliable interpretation tree. Currently, the interpretation trees are represented by semi-automatically written Lisp programs. We plan to develop an AI program which generates interpretation trees represented by object-oriented programming.

2.2 Frame-Based Representations for Geometrical Reasoning in Vision

Walker has been working on a scheme for representing and reasoning about geometrical objects, such as projections between 2D images and 3D scenes, shape and surface properties of objects, and geometrical and topological relationships between objects. These capabilities are essential for knowledge-based, 3D photo-interpretation systems which combine domain knowledge with image processing, as demonstrated by such systems as 3D MOSAIC [6] and ACRONYM [2].

We adopt a frame-based representation using the CMU-built Framekit tool in Common Lisp. Each type of object, such as a point, line, or plane, is represented as a frame. Each frame contains slots for parameters of the object, such as the equation of a line and points that lie on the line. In addition, demons are associated with the slots to invoke appropriate functions if information is added to, deleted from, or needed from the slot. For example, when a new point is added to the list of points on a line, the current line equation is deleted, and later when the line equation is needed, it is computed by least squares fitting to the set of points. Specific objects are created by instantiating the generic frame. Instantiating an object consists of creating a new frame with a unique name and filling in slots specific to the new object. Slots not filled in are inherited from the generic object by means of an IS_A hierarchy.

Geometric relationships between objects are also represented by frames. A few examples of primitive relationships are:

- For line-line
    - PERPENDICULAR
    - PARALLEL
    - INTERSECTING
- For line-plane
    - LINE-ON-PLANE

Each geometric relationship has slots for two or more geometric objects plus one or more numeric ranges. In addition, there is a **compute** slot which is never filled, but contains a demon to evaluate the relationship. The evaluation function for a relationship first attempts to fill in any missing slots, either hypothesizing a geometric entity or computing a numeric range. Then information from the relationship is added to each geometric entity. For example, the LINES-IN-PLANE relationship adds each line to the plane's list of contained lines and adds the plane's normal to the list of vectors perpendicular to each line. Finally, computations are done to ensure that the true numeric values (such as the angle between two lines) fall within the specified ranges. If the values fall outside the ranges, then the evaluation function returns FALSE, indicating an inconsistency in the data.

The above primitive geometric relationships are combined into conjunctions to describe the complex geometric relationships between objects. One slot of each conjunction contains a description of the relationships that will be evaluated. The other slots contain "variables" for the primitive relationships. Each slot of a primitive relationship may contain a reference to a slot of the conjunction, a reference to a local variable, or a constant. When the conjunction is evaluated, it instantiates and evaluates each of the primitive relationships in turn, using the specified slot and local values. the changes made by the primitive relationship (new hypotheses, or updates to existing hypotheses) are stored in the appropriate conjunction slots and local variables, and the next relationship is instantiated.

To perform geometric reasoning using this framework, a conjunction is built for each concept and the slots corresponding to known information are filled in. Successful evaluation of the conjunction results in hypotheses for the remaining slots of the concepts. This model has been used to define such concepts as a roof and a wall of buildings for city scene understanding, a similar task domain of 3D MOSAIC [6] in aerial photo interpretation. With simulated input, running these conjunctions has generated correct hypotheses for missing edges and vertices of rectangular buildings. We plan to incorporate this framework of geometrical representation and reasoning into an image interpretation system by means of an image processing interface and a hypothesis tester.

3. Color Understanding

Using the facilities of the Calibrated Imaging Laboratory [11], whose recent progress is presented in **6**, we have been developing techniques for using color information in a manner which is sound computationally and physically. Current work includes measurement of gloss components from color images and extraction of color edges.

3.1 Gloss from Color

Klinker, Shafer, and Kanade have worked on the measurement of gloss from color images. All of the image segmentation methods that are widely used today are confused by artifacts such as highlights, because they are not based on any physical model of these phenomena. We have developed and implemented a method for automatically separating highlight reflection from matte object reflection. By exploiting the color difference between object color and highlight color, our algorithm generates two intrinsic images from one color image of a scene, one showing the scene without highlights and the other one showing only the highlights. The successful modeling of highlight reflection can provide a useful preprocessor for stereo and motion analysis, for direct geometric shape inference, for color image segmentation, and for material type classification.

Our work is based on a spectral theory of light reflection from dielectric materials [12]. The theory describes the color at each point as a linear combination of the object color and the highlight color. According to this model of light reflection, the color data of all points from one object forms a planar cluster in the color space. The shape of the cluster is determined by the object and highlight colors and by the object shape and illumination geometry. We use the shape of such clusters to determine the amount of highlight reflection and matte object reflection at each image point. This method has been successfully run on several real images. The article "Using a Color Reflection Model to Separate Highlights from Object Color" (by Klinker, Shafer, Kanade) in this volume contains example results from running the program on the real image.

3.2 Color Edge Detection

Novak and Shafer have been studying color versions of the Canny edge detection operator, and have obtained both theoretical and pragmatic results. In general, the color edges are noticeably better than edges from intensity images as evaluated by human judgement. The basic Canny operator can be briefly described as running x- and y-derivative operators over a smoothed image to yield the quantities I_x and I_y; from these, the magnitude and direction of the best edge can be found. In a color image, the pixel value is the vector $C=[R\ G\ B]$, and the gradient operator per se no longer applies. Rather, we have C as a function of (x,y) in the image, and can describe its variation at any point by

$$\Delta C = J\Delta(x,y)$$

where J is the Jacobian matrix containing the derivatives of each color band:

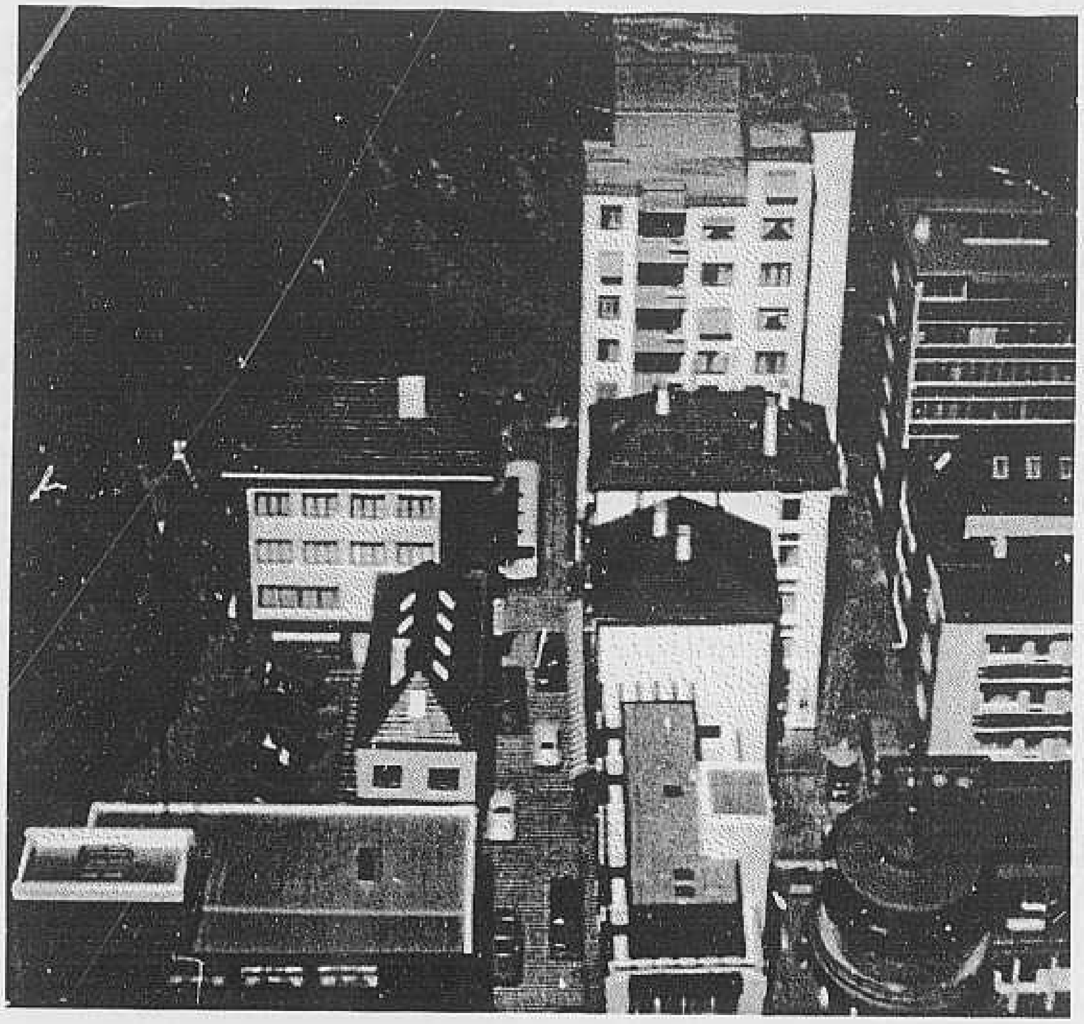


Figure 2: A sample image in the set used for evaluating color edge operators

$$J=\begin{bmatrix} R_x & R_y \\ G_x & G_y \\ B_x & B_y \end{bmatrix}=[C_x\ C_y]$$

The direction of travel in the input (image) that produces the greatest magnitude change in the output (color) is given by the eigenvector of J^TJ that corresponds to the largest eigenvalue, and the magnitude of the output change will be the square root of that eigenvalue. This solution would hold for other multivariate edge-finding methods as well, such as finding color/texture edges in x–y–t image sequences. In the case of color edges of a single image, the solution is that the edge angle θ is given by:

$$tan2\theta=\frac{2C_x \cdot C_y}{\|C_x\|^2-\|C_y\|^2}$$

where C_x and C_y are the partial derivatives of the color, e.g.

$$C_x=[R_xG_xB_x]^T.$$

The magnitude m of the edge will be given by:

$$m^2=\|C_x\|^2cos^2\theta+2C_x\cdot C_y sin\theta\, cos\theta+\|C_y\|^2sin^2\theta$$

The above method can be described in terms similar to the original Canny operator. First, the x- and y-derivatives of each color band are calculated independently after smoothing the images. Next, the above formulae are used to calculate the magnitude and direction of each edge. Finally, local non-maximum suppression is applied to eliminate "broad" edges.

The above method is theoretically sound, but as a practical method it is computationally expensive. We have also developed a less expensive, general method for creating color operators from multi-stage intensity operators, that is from operators with several steps of processing. This method involves numbering the steps from 1 to n; then, a color operator at stage k can be derived by carrying out steps 1 through k independently on each color band, combining the results with a combination operator, and performing steps k+1 through n on the single-valued result. The combination operators are typically color-distance operators such as the L_1 norm (sum of the absolute values), L_2 norm (Euclidean distance), or L_∞ norm (maximum absolute value). The color operator can then be characterized by the stage k at which combination takes place, and the index i of the L_i norm used for combination. Thus, for the Canny operator the stages can be numbered as (I) calculate directional derivatives, (II) calculate edge magnitude and direction, and (III) perform non-maximum suppression. The I/2 Color Canny operator would then consist of finding directional derivatives of each color band independently (stage I), combining these by a Euclidean color distance metric (L_2) to form the total magnitude of the color change in x and y, and performing the rest of the calculation using these values. We note that 0/1 and 0/2 color operators consist of calculating an intensity image (using a sum or sum-of-squares) and evaluating the standard Canny operator; also, note that the I/1 operator is the same as the 0/1 operator since the directional derivatives are calculated by a linear transform from the image. Finally, we note that a stage IV color operator consists of evaluating edges separately (stages I-III) on each color band, then combining the resulting edge images.

We have evaluated the theoretical color Canny operator and all of the non-trivial multi-stage color operators on a set of images of our landscape model. Figure 2 is a sample image in the set: this scene is quite complex. By visually comparing these operators' output with each other and with the output of the Canny operator applied to intensity image of these scenes, we have the following conclusions for this set of images:

- The color edges are consistently better than edges from intensity images, though most (over 90%) of the edges are about the same.
- The best multi-stage operator seems to be II/∞, that is, calculating the magnitude and direction independently for each color band and then selecting the edge with the maximum magnitude.
- A similar result, not quite as good, was obtained from the I/∞ operator. This is faster than the II/∞ operator since the color combination is performed earlier.
- The II/∞ operator produced almost exactly the same edges as the theoretical operator based on the above Jacobian analysis.
- The image input quality was very important; producing better quality input images yielded better results. Surprisingly, images digitized off of an NTSC encoded color signal produced color edges almost as good as those from direct R-G-B color digitization.

We have applied similar lines of reasoning to develop color operators for stereo feature point detection and matching, and have achieved a great reduction in the matching error rate using color. Our future work in this area will include developing "smarter" color difference operators that can take into account camera and lighting properties and may be adaptive to local conditions in the image or scene.

4. Modeling Uncertainty in Representations for Low-Level Vision

Research on obtaining and representing scene information from images has been one of central topics of the CMU Image Understanding program. Recently Szeliski has been working on modeling uncertainty in low-level dense representations, such as depth maps and optical flow (velocity) maps by means of Bayesian model. The Bayesian modeling has been already used in low-level vision processing by other researchers. However, one of the distinguishing features of Szeliski's approach is that he uses Bayesian modeling not only to recover optimal estimates (as is currently done), but also to calculate the uncertainty associated with these estimates.

Low-level representations are usually derived from the input image(s) using "shape-from-X" methods such as stereo or shape from shading. These methods usually yield data that is sparse (e.g. stereo) or underconstrained (e.g. shape from shading). To overcome this problem, and to obtain a full-field map, two approaches are currently popular. The first, called regularization [10], reformulates the problem in terms of the minimization of an energy functional. Smoothness constraints, in the form of added energy terms, are used to guarantee a unique and well behaved solution. The second method, Bayesian estimation [15], assumes a probabilistic prior model for the data being estimated, and a probabilistic imaging model relating the data to the sensed image. An optimal estimate (e.g. Maximum A Posteriori or Minimum Variance) can then be obtained.

One of the results of this research has shown that regularization methods are equivalent to Bayesian models with *fractal* priors, i.e. models that are self-similar over scale transformations. Suppose that depth constraints (eg., sparse measurements of depth) are given as Figure 3(a). A *typical* sample of a likely map (Figure 3(c)) thus has the rough (crinkly) appearance of a fractal, as opposed to the *most likely* sample, which is maximally smooth (Figure 3(b)). This result also leads to a new fractal generation algorithm based on the multi-resolution stochastic (Monte Carlo) simulation of Markov Random Fields. This allows the use of arbitrary constraints (e.g. depth values, orientation or depth discontinuities) without affecting the fractal nature of the resulting surface.

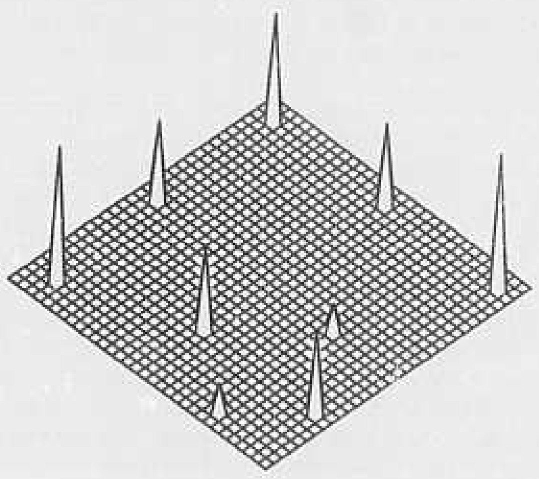
The main emphasis of this research, however, is to study how the uncertainty inherent in the Bayesian modeling approach can be estimated and used in further processing. Previous work, both in regularization and Bayesian estimation, has concentrated solely on obtaining a single optimal estimate of the underlying field. However, the Bayesian approach actually (implicitly) defines a whole distribution conditional on the sensed data. For example, when regularization is used, the resulting distribution is a multivariate correlated Gaussian.

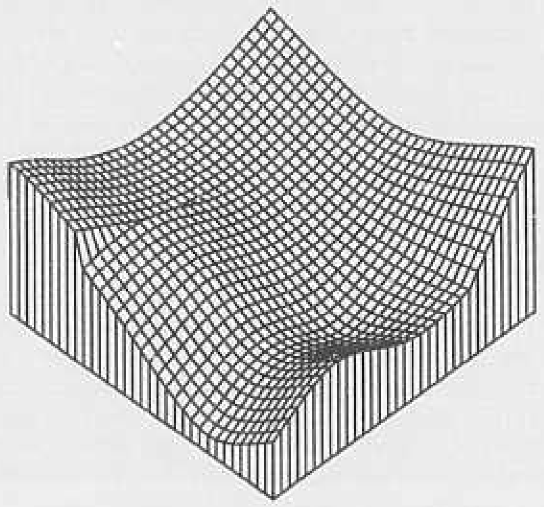
Thus knowing both the mean (minimum variance estimate) and covariance fully characterizes the distribution. Estimating the variance or covariance can be done either by using a deterministic technique (which is slow) or by using stochastic (Monte Carlo) simulation. The estimated uncertainty (Figure 3(d)) can then be used for further processing, such as integration with new data, or matching to a model. Current research is focusing on the former application (using Kalman filtering), as well as examining the use of alternate representations that better model the uncertainty.

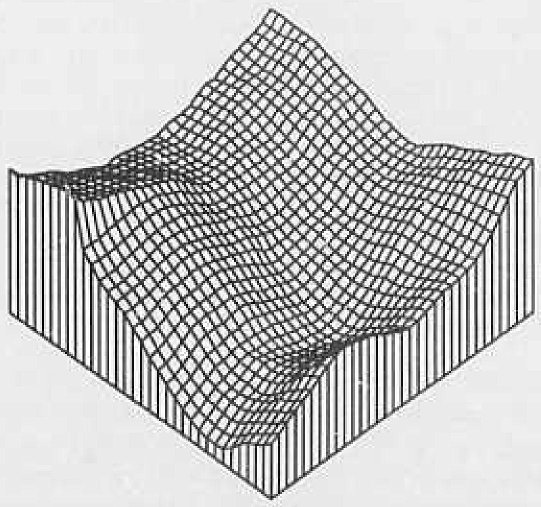
5. Parallel Vision on Warp

Warp is the Carnegie Mellon Systolic Array Machine providing 100 MFLOP. As part of Strategic Computing Vision, Webb and his associates have been developing vision software for use by vision researchers [1]. To date, we have achieved the following:

- Several demonstrations of Warp's use for road following, obstacle avoidance using stereo vision and ERIM laser range scanner data, NMR image processing, signal processing, and other vision algorithms.


(a) Original depth contraints


(b) Minimum energy solution (thin-plate model)


(c) Typical solution (fractal)

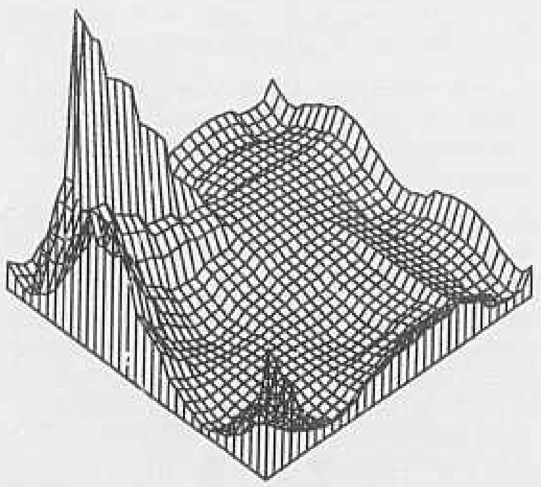

(d) Variance field (ie., diagonal elements from the covariance matrix

Figure 3: Bayesian modeling of a surface

- A library based on the SPIDER FORTRAN subroutine library, all written in the Warp programming language (W2). The current library includes about 80 different Warp programs, covering edge detection, smoothing, image operations, Fourier transform, and so on. The actual number of routines in the SPIDER library covered by these Warp programs is about 100.

Secondly, software tools for facilitating *vision* programming on Warp are being developed on top of the generic W2 Compiler of Warp. Hamey, Webb, and Wu [4] have developed a special-purpose programming language, called Apply, in which low-level vision (local operation) programs can be written quickly and efficiently. By simply describing the local operations on a local window, the Apply compiler can generate codes for the Warp machine (in W2) which execute the operations on the whole image efficiently. The compiler can also generate codes in C under Unix, which allows debugging algorithms off Warp.

Another important development around Warp is that as the software environment improves, it is becoming a tool for vision *research* (not for demonstrations of architectural concepts) in our CMU Image Understanding group. For example, for his research on for analyzing repetitive textures, Hamey needed to detect local point symmetry to locate the texture elements. Point symmetry is detected by an Analysis of Variance (ANOVA) statistical test which is applied to a window surrounding each pixel location. The ANOVA method consists of partitioning the variance of the data into two portions: that which is explained by the model and that which remains unexplained. The method is to be applied at each pixel location to measure point symmetry. Local peaks in an image of a symmetry measure values represent points of local symmetry. This analysis requires a large amount of computation.

The Warp implementation of this algorithm performs 346 million multiplications and 519 million additions. The prototype Warp processes a 512×512 image in 30 seconds. The same processing would take more than an hour on a SUN-3.

6. The Calibrated Imaging Laboratory

In the last year Shafer completed the initial development of the Calibrated Imaging Laboratory (CIL), as described in 1985 IU Workshop Proceedings [11]. This laboratory is a small room providing fairly precise control of the illumination, background reflection, geometry, and color conditions for imaging. The key features of the CIL are:

- *Flexible lighting control*, including a "point source" accurate enough for shadow edge studies.
- *Geometric control of the camera and object positions*, to allow for controlled position, motion, and stereo configurations.
- *Geometric measurement* by theodolites (surveyor's transits) with a "geometric calculator" program that allows 3D scene points to be measured and their pixel locations calculated to the nearest pixel in the image.
- A *variety of cameras* including sets of color and other filters, RGB color cameras, and a high-precision camera yielding 512x512x8-bit images that are nearly noise-free.
- A *variety of test objects* including calibration materials, simple objects for color and texture studies, and a highly detailed landscape model for studying images of a complex environment within the laboratory.

We are currently working on the geometric camera calibration methodology for the CIL, which should achieve higher precision than current methods, and which will involve *controlling* the fine motion of the camera to put the camera into a standard orientation relative to the lab; and we are implementing software for the control of the high-precision camera so we can utilize its images more effectively. We recently acquired an inexpensive R-G-B color camera and will be evaluating its spatial and color resolution, and we plan this year to motorize our camera positioning mount and to acquire a spectroradiometer for measuring the spectrum of narrow beams of light. Our goal is to provide images with every bit noise-free and with ground truth data that allows any pixel value to be exactly calculated from direct measurements of the scene; we expect to achieve this goal in 1987.

The CIL has provided data already for several vision projects, including our studies of color and highlights, color edges, motion, and image segmentation. We have also provided tapes of images for other universities, and we have provided assistance for other labs in deciding what equipment to obtain, such as cameras and color filters. We have a series of lab reports (CILIA) that describe the facilities of the lab and the issues in acquiring and using this kind of equipment.

References

[1] Annaratone, M., Bitz, F., Deutch, J., Hamey, L., Kung, H. T., Maulik, P., Ribas, H., Tseng, P. and Webb, J.
Applications Experience on Warp.
In *Proceedings of the 1987 National Computer Conference*. AFIPS, 1987.

[2] Brooks, R. A.
Symbolic Reasoning among 3-D Models and 2-D Images.
AI 17:285-349, 1981.

[3] Goto,Y., Matsuzaki,K., Kweon,I., Obatake,T.
CMU Sidewalk Navigation System.
In *Proceedings 1986 Fall Joint Computer Conference*, pages 105-113. November, 1986.

[4] Hamey L., Webb, J., and Wu, I.
Low-Level Vision on Warp and the Apply Programming Model.

Technical Report To Be Published, Carnegie-Mellon University Computer Science Department, 1987.
Submitted for publication to "Parallel Computation and Computers for Artificial Intelligence", Edited by Janusz Kowalik, Published by Kluwer Academic Publishers.

[5] Hebert, M., and Kanade, T.
Outdoor Scene Analysis Using Range Data.
In *IEEE International Conference on Robotics and Automation*. 1986.

[6] Herman, M. and Kanade, T.
The 3D MOSAIC Scene Understanding System: Incremental Reconstruction of 3D Scenes from Complex Images.
In *Proc DARPA Image Understanding Workshop*, pages 137-148. 1984.

[7] Stentz, K.
Precompiling a Geometrical Model into an Interpretation Tree for Object Recognition in Bin-Picking Tasks.
In *(in this volume)*. 1987.

[8] Kanade, T., Thorpe, C., and Whittaker, W.
Autonomous Land Vehicle Project at CMU.
In *Proc. ACM Computer Conference*. Feb, 1986.

[9] Takeo Kanade and Charles Thorpe.
CMU Strategic Computing Vision Project Report: 1984 to 1985.
Technical Report, The Robotics Institute, Carnegie-Mellon University, 1985.

[10] Poggio, T., Torre, V., and Koch, C.
Computational Vision and Regularization Theory.
Nature 317(6035):314-319, September, 1985.

[11] Shafer, S. A.
The Calibrated Imaging Lab Under Construction at CMU.
In *Proc DARPA Image Understanding Workshop*, pages 519-515. 1985.

[12] Shafer, R.A.
Using Color to Separate Reflection Components.
Color Research and Application 10(4 (Winter)):210-218, 1985.

[13] Shafer, S., Stentz, A., Thorpe, C.
An Architecture for Sensor Fusion in a Mobile Robot.
In *IEEE International Conference on Robotics and Automation*. 1986.

[14] Stentz, A. and Goto, Y.
The CMU Navigational Architecture.
In *(in this volume)*. 1987.

[15] Szeliski, R.
Cooperative Algorithms for Solving Random-Dot Stereograms.
Technical Report CMU-CS-86-133, Carnegie-Mellon University, June, 1986.

[16] Thorpe, C., Shafer, S.A., and Kanade, T.
Vision and Navigation for the Carnegie Mellon Navlab.
In *(in this volume)*. 1987.

[17] Wallace, R., Matsuzaki, K., Goto, Y., Crisman, J., Webb, J., and Kanade, T.
Progress in Robot Road-Following.
In *IEEE International Conference on Robotics and Automation*. 1986.

MIT PROGRESS IN UNDERSTANDING IMAGES

T. Poggio and the Staff

The Artificial Intelligence Laboratory
Massachusetts Institute of Technology

ABSTRACT

Until recently our work has focused primarily on the initial processes and representations of early and middle vision that decode information about 3-D surfaces and their properties. The emphasis of our work is now centered on the integration of different sources of information and on object recognition. We will discuss recent progress in texture, stereo, motion, then outline work in middle vision concerning the integration of depth with intensity data and finally sketch several approaches to recognition. We will also describe our Vision Machine project – an integration testbed for our vision research – and our mobile robot – an experiment in autonomous navigation.

1. INTRODUCTION

Our present work spans early and high level vision. This report describes some of the results of the past year, from edge detection to recognition. Much of the present emphasis in early vision is focused in developing parallel and robust algorithms on the Connection machine. The Connection machine is part of the Vision machine system – our testbed for integrating early vision modules and, later, recognition algorithms. Our research on the integration of early vision modules is mostly based on an extension of regularization theory, that builds upon Markov Random Field models. Recognition is a multifacet problem: we describe several approaches to it. Recognition is not the only goal of vision: navigation is another important task that relies heavily on vision capabilities. Therefore, we also describe our work in mobile robot vision.

2. EARLY VISION

We have described in our previous reports our work in regularization theory – a theoretical framework that
unifies several solution to early vision problems and i
based on their ill-posed nature. Our work on developin
and applying the theory has continued: we describe her
some of the results. We have also done further work i
the use of texture information and in developing stere
and motion algorithms.

2.1. Finding the Optimal Scale

Geiger and Poggio have considered the problem of finc
ing the optimal regularization parameter λ – corr
sponding to the scale of the filter – in the case of edg
detection. They derive an optimal filter that is moı
general than the one computed by Poggio, Voorhe
and Yuille (1984). They derive a formula relating tl
signal-to-noise ratio to the parameter λ from regula
ization analysis, showing that the scale of the filter
a function of the signal-to-noise ratio. An implement
tion of their scheme has been shown to work on natur
images and even to explain two perceptual phenomen
coarsely quantized images become recognizable by
ther blurring or adding noise.

2.2. Texture for Labeling Edges

Texture provides a cue for image segmentation, sin
texture boundaries are often due to surface discontin
ties in the scene. The paper in these Proceedings l
Voorhees and Poggio describes a token-based meth
for detecting texture boundaries in images of nat
ral scenes. According to this theory, texture is repı
sented by small-scale intensity features, such as ele
gated blobs, that represent a simple subset of Jule
"textons". Texture boundaries are identified as plac
where there are sharp differences in the attributes
these textons, such as density, size and orientation. I
terestingly, a smaller set of tokens than the one su
gested by Julesz is sufficient to account for texture d
crimination: the reason is the initial filtering stage tl

is neglected by Julesz in his theory. Textures that he believed can only be discriminated on the basis of number of crossings or terminations become distinguishable on the basis of contrast or other simple blob attributes (width, etc.) after convolution of the image with a filter such as the laplacian of a Gaussian.

Voorhees and Poggio have developed and implemented a scheme to extract blobs as texture tokens from natural images, an important step which has been ignored in most of the psychophysical theories of human texture discrimination. An improved noise estimation scheme is employed as part of this step. Once textons are extracted, potential texture boundaries are identified by detecting discontinuties over first order statistics of attributes of neighborhoods of textons. These boundaries are then verified using statistical hypothesis testing.

2.3. Stereo

Binocular stereo is one of the most precise sources of depth information. In the past we have reported on our work on the Marr-Poggio stereo algorithm and its further development and application to aerial stereo photos by Grimson. We describe here a parallel implementation of a new stereo algorithm on the Connection machine. The stereo algorithm is related to schemes proposed by Marroquin (1983), Prazdny (1985), and especially Marr and Poggio (1976). These algorithms are feature based. We have recently explored an algorithm based on matching intensities directly. It is conceivable that a similar scheme may complement a feature based algorithms to provide a denser depth map.

2.3.1. Parallel Stereo and the Forbidden Zone Constraint

A new simple but fast algorithm was implemented in collaboration with Thinking Machine corporation on the TMC Connection Machine (Drumheller and Poggio, 1986). Some of its features are: a) the potential for combining different primitives, including color information, b) the use of a stronger and new formulation of the uniqueness constraint and c) its disparity representation that maps efficiently into the CM architecture.

The algorithm consists of the following steps:

Compute features for matching.

Compute potential matches.

Determine the amount of local suport for each potential match.

Choose correct matches on the basis of local support and constraints on uniqueness and ordering.

The algorithm does not require a particular type of matching feature. Different types of features can be used, such as the sign of the convolution with a difference of gaussians. Let us assume that the images are perfectly registered and that all epipolar lines are horizontal. We can represent the set of potential matches in a *one-dimensional* stereomatching problem (i.e., for a pair of epipolar lines) by the same diagram used by Marr & Poggio (1976). This representation of the stereo problem is the starting point for mapping a stereo algorithm into the CM.

Potential matches are allowed to occur between two zero crossings of the same sign. This implements the *compatibility constraint* (Marr & Poggio 1976). The first step at distinguishing the correct matches from the false ones is to apply the *continuity constraint* (Marr & Poggio 1976). This principle states that since most surfaces in the real world are piecewise smooth, potential matches should be selected that result in a piecewise smooth disparity function. A straightforward way to measure how well each disparity satisfies the smoothness condition is to convolve the three-dimensional region of x-y-d-space contained by the field P with a three-dimensional kernel that gathers suppotrs from smooth configurations of potential matches. There are many different kernels, or *support functions*, that will do a good job on this task. Marr & Poggio (1976) uses a very simple support function (or "excitatory region") that is circular, uniformly-weighted, and flat, i.e., it occupies only one level in the disparity dimension. Every potential match is surrounded by an hourglass-shaped *forbidden zone*. In the forbidden zone there must be *no more than one match*, unless the scene contains transparent or narrowly-occluding objects. Examples of such special scenes include a pane of glass with markings on both sides or a vertical wire suspended in front of a textured wall. These situations violate the *ordering constraint* (Yuille & Poggio 1984). If we assume that the scene contains only opaque objects, with no narrow occlusions, then it makes sense to enforce uniqueness not only along line of sights, but along *any line of sight in the forbidden zone*. A correct match should be the only match in its entire forbidden zone. The Marr-Poggio algorithm and the winner-take-all algorithms mentioned earlier (Prazdny 1985, Pollard, Mayhew & Frisby 1985, Marroquin 1983) use only the left and right eye lines of sight, which comprise a small subregion of the entire forbidden zone. Notice that the forbidden zone property is reflexive: if a match lies in the forbidden zone of another one, the latter is in the fobidden zone of the first (Yuille and Poggio, 1985, where transitivity is also proven).

The algorithm enforces uniqueness by suppressing all matches that lie in the forbidden zone of the match that gathers maximum support from the 3-D convolution operation. The process of non-maximum suppression along lines of sight is called the *winner-take-all* approach; it is analyzed in detail in (Marroquin 1983). Drumheller and Poggio use a stronger, more general version of the winner-take-all method. Instead of applying non-maximum suppression only along the left and right eye lines of sight, they applied it across the entire forbidden zone. In general, the use of the entire forbidden zone in the winner-take-all step results in fewer matches than with just the left and right eye lines of sight. However, the number of errors almost always decreases more than the number of matches, especially in the occluded region. Therefore the ratio of errors to matches decreases. This supports the hypothesis that the entire forbidden zone could be exploited to advantage for scenes known to contain only opaque objects with no narrow occlusions.

Drumheller and Poggio have also used very simple statistics of the voting process (how close the winners are and where ties are) to obtain some initial information about depth discontinuities. Preliminary experiments indicate the fesibility of the method.

It turns out that the stereo algorithm described here is a specific instance of a new regularization method called "constraint method' that we have also applied to motion (see later). The stereo algorithm runs on the Connection machine system with good results on natural scenes in times that are typically of the order of one second.

2.3.2. Brightness-Based Binocular Stereo

Gennert and Horn have developed a new brightness-based stereo matching method that does not rely on correlation. Instead, a spatially varying linear transformation is used to relate grey-levels in the two images, so that the matching criterion is relaxed. The problem of stereo matching then reduces to solving for the parameters of this linear transformation, which is allowed to vary slowly from point to point in the image. The method has been implementation on a highly parallel computer, the Connection machine system, and several test cases run. Run time is about 6-8 seconds for a 128x128 image.

There are two main attractions of feature-based methods. First, features that can be reliably detected may be used. In general, scene characteristics that give rise to features in one image will tend to produce similar features in another image. That is, features tend to be robust, especially with respect to small changes in viewing direction, which is exactly what is necessary to perform reliable stereo matching. By contrast, grey-levels at corresponding points in two images may differ significantly. Changes in photometry due to changes in viewing direction, sensor noise and differences in the calibration of the two cameras conspire to virtually guarantee that few scene points will give rise to the same brightness values in the two images. Secondly, matching combinatorics are reduced by using features rather than image grey-levels as match primitives. The number of features is usually much smaller than the number of pixels, therefore the reduction in effort accompanying a decrease in the number of possible matches is substantial.

Feature-based methods, however, also have a number of draw-backs. Perhaps the most significant one is that the depth information is sparse, and detailed surface information has to be generated by interpolation, essentially a process of educated guessing based on some model of what the class of surfaces of interest should look like. Information is not generated in areas where image contrast at high spatial frequencies is insignificant. Therefore, intensity based schemes could be used to fill-in between matched features.

Michael Gennert has developed a linear transformation model of image brightness matching and has illustrated its applicability to the stereo problem. The transformation consists of spatially-varying fields of gain and offset that relate the grey-levels in the two images. This contrasts with earlier brightness matching work in that image correlation is not used, and it is not required that conjugate image points have the same image brightness. The variational approach to image matching as discussed by Horn (1986) does not work robustly on real image data as it stands. What happens is that iterative solutions of the variational problem tend to get trapped in local extrema unless the initial guess of the disparity field happens to be almost correct. Gennert has solved this problem in the obvious way by working at multiple scales; using the coarse solutions as initial conditions for the more detailed ones.

Recently, this new approach has been applied to other problems besides stereo. In particular, Gennert and Negahdaripour have shown how it can be used to obtain more robust parallel algorithms for estimating the optical flow. In previous work on this problem by

Berthold Horn and Brian Schunck (see Horn, 1986) it was assumed that the brightness of a patch does not change as it moves. In the new approach a small change is permitted, as long as it is consistent with the changes in the neighborhood of the point under consideration. This method also has been implemented on the Connection Machine.

2.4. Motion

Until now work on the computation and use of motion information has been mostly theoretical: little has been done with motion sequences of real images, because of the limitations in computational power. A consequence of this state of affairs is that little attention was given so far at numerical stability of algorithms and at validity of underlying assumptions for real scenes. Horn and Negahdaripuor have addressed the first problem by developing algorithms that are tailored to restricted types of motions and/or surfaces. Verri and Poggio question the usual assumption used in many motion schemes, that the optical flow is a close quantitative measure of the motion field to be used for computation of structure from motion. They suggest a different, more qualitative use of the optical flow.

2.4.1. Motion Relative to a Planar Surfaces

Some autonomous vehicles will have to deal with visual information that comes in part from a planar or nearly planar surface such as a landing field, a road or parts of the ocean floor. Analysis of time-varying images provides information both about the vehicle motion and the position and orientation of the surface. Horn and Negahdaripuor have developed an iterative algorithm as well as a closed form solution for recovering the motion of an observer relative to a planar surface directly from image brightness derivatives. They do not estimate the optical flow as an intermediate step, using only the spatial and temporal brightness gradients. They solve a linear matrix equation for the elements of a 3×3 matrix. The eigenvalue decomposition of its symmetric part is then used to compute the translational and rotational motion parameters and the orientation of the plane.

The problem of recoverying rigid body motion and surface structure uniquely from image data has been the topic of many research papers in the area of machine vision. Many approaches based on matching feature points, tracking contours, and using velocity flow field, texture, or image brightness gradients have been proposed in the literature, but not many have addressed the important issue of uniqueness.

Feature point matching schemes require the detection of local brightness patterns that are likely to be found in consecutive images. A correspondence problem between the elements in successive 2-D images has then to be solved in order to allow the 3-D motion and spatial configuration of these isolated features to be recovered. The minimum number of points required to recover the 3-D motion uniquely depends on the number of image frames. With two frames, in most cases, a minimum of 5 points results in a unique solution from a set of nonlinear equations. It turns out that if one uses more points, namely 8, as in algorithms proposed by Longuet-Higgins, Tsai and Huang, Buxton et al., one only needs to solve linear equations. (Unfortunately these linear methods are not at all robust.) In any case, all of these methods require feature detection and matching and fail to give reliable results when the object in view is smooth with no well-defined features. Further, since these methods use information only from a small portion of the image, they are noise sensitive. When nearby feature points are selected, these methods become even more sensitive to small amounts of error in the data.

For smoothly curved surfaces, Longuet-Higgins and Prazdny suggested a method that uses the optical flow and its first and second derivatives at a single point. They reduced the problem to that of solving a cubic equation and concluded that, in general, three solutions are feasible. Later, Waxman and Ullman developed the method proposed there into an algorithm for recovering the structure and motion parameters from a set of nonlinear equations. They had to treat many special cases, and uniqueness results were shown only through numerical examples. More recently, Waxman et al. found a closed form solution to the original formulation. These methods are very noise sensitive since second order derivatives of errorful optical flow data are used. More robust algorithms that use the information from the whole region of the image plane have been suggested, but they still require the computation of a dense flow field from a sequence of images.

The flow-field based approaches assume that a reasonable estimate of the optical flow is available. In general, the computation of the local velocity field assumes thee validity of a constraint equation between the local brightness changes and the two components of the optical flow. Even when this is true, it only gives the component of the velocity field in the direction of the brightness gradient. To compute the full field, one needs additional constraints such as the heuristic assumption that the flow field is locally smooth. In many cases, this leads to optical flow fields that are not consistent with the true velocity field. Since, in addition, velocity based approaches to structure from motion are not robust, i.e., the solution may change drastically with only a small amount of noise in the data, one can argue that approaches that use an optical flow field that is computed through heuristic assumptions are apt to fail.

Shariar Negahdaripuor and Berthold Horn have developed a method that uses the brightness values in a sequence of images directly, in order to recover the motion parameters as well as the local structure of the surface patches on the object. Since they do not estimate the optical flow, they do not need to make any heuristic assumptions (apart the underlying constraint equation). They assume that the surface is textured or has surface markings, and that it is smooth so that it can be approximated in a local region by a planar patch. They give a closed form solution for the motion and surface parameters, and show that there can be only two solutions.

2.4.2. Qualitative Properties of Optical Flow

Verri and Poggio have shown (these Proceedings) that the *optical flow*, the 2-D field that can be associated with the variation of the image brightness pattern, and the 2-D *motion field*, the projection on the image plane of the 3-D velocity field of a moving scene, are in general different, unless very special conditions are satisfied. The reason is that the constraint equation can only be satsisfied in special cases. The optical flow, therefore, is ill-suited for computing structure from motion and for reconstructing the 3-D velocity field, problems that require an accurate estimate of the 2-D motion field. As a consequence, Verri and Poggio argue that the optical flow should be used to yield information of a more qualitative type. The optical flow (and even the so-called normal component of the optical flow) are very useful, for instance, for computing a fast, initial estimate of motion discontinuities (see for instance Poggio and Reichardt, 1983). Other qualitative properties of the 2-D motion field can give useful information about the 3-D velocity field and the 3-D structure of the scene, and can be usually obtained from the optical flow. To support this approach Verri and Poggio show how the (smoothed) optical flow and 2-D motion field, interpreted as vector fields tangent to flows of planar dynamical systems, may have the same *qualitative* properties from the point of view of the theory of structural stability of dynamical systems.

2.4.3. A Parallel Constraint Algorithm for Optical Flow

J. Little, H. Buelthoff and T. Poggio (these Proceedings) have developed a new, parallel and fast algorithm for computing the optical flow. The algorithm is based on a new regularization method that we call "constraint method". The method, based on a theorem of Tikhonov, can enforce local constraints and lead directly to efficient, parallel algorithms. The specific constraint exploited by our algorithm can be shown to correspond, in its most general form, to 3-D rigid motion of planar surfaces. Segmentation of the motion field can be obtained from the optical flow field generated by the algorithm. An iterative scheme provides fast, approximate solutions and refines them subsequently. The algorithm has been implemented on the CM and demonstrated in real-time processing tasks as part of the Vision Machine system.

2.4.4. Robust Algorithms for Structure from Motion

Ullman (1984) recently proposed an algorithm for recovering 3–D structure from motion that integrates image motion information over an extended time, and interprets both rigid and nonrigid objects in motion. The algorithm uses the *rigidity* constraint in a more flexible way than previous techniques. The algorithm, called the *incremental rigidity scheme*, maintains an internal model of the structure of a moving object, which is continually updated as new positions of image elements are considered. The initial model may be flat, if no other cues to 3–D structure are present, or it may be determined by other cues available, for example, from binocular stereo, shading, texture or perspective. As each new view of the moving object appears, the algorithm computes a new set of 3–D coordinates for points on the object that maximizes the rigidity in the tranformation from the current model to the new positions. In particular, the algorithm minimizes the change in the 3–D distances between points in the model. Ullman's original formulation assumes the input to be a sequence of discrete frames, each containing a set of discrete feature points. Through the process of repeatedly considering a new frame in the sequence and updating the current model of the structure of the features, the scheme builds up and maintains a 3–D model, and can be applied to both rigid and nonrigid objects in motion. This scheme has a number of computational advantages: (1) by integrating information over an extended time, it provides a stable recovery of structure in the presence of error in the image measurements, (2) it allows deviations from rigidity, while always maintaining some model of 3–D structure, (3) it allows interactions with other sources of 3–D information, and (4) empirical studies suggest that the algorithm is able to recover the correct 3–D structure, when begun with a flat initial 3–D structure that is typically very different from the true object structure.

We have developed Ullman's work further, in several directions (Hildreth and Grzywacz, 1986). First, we developed a continuous formulation of the incremental rigidity scheme that uses velocity information at discrete points as input to the recovery process. Through computer simulations and a theoretical analysis, we examined the behavior of the discrete and continuous formulations as a function of the angular displacement between frames. We also developed discrete and continuous formulations that use perspective projection, and explored the behavior of the perspective formulations through computer simulations. We have recently integrated our implementation of the incremental rigidity scheme with simple schemes for tracking localizable features through image sequences, and tested the algorithm with natural motion sequences.

The main conclusions of this work are the following. The direct use of velocity information as input to the incremental rigidity scheme provides a rough estimate of 3–D structure over a short viewing period, but does not allow a robust recovery of structure over an extended time. The computation of a stable long–term solution requires the use of views of a moving object that differ significantly. This implies the need for a recovery process with memory of past views, but this memory need not be extended indefinitely and continuously into the past. A small number of discrete views are sufficient for recovering 3–D structure if they differ significantly from one another. For rigid objects rotating about a fixed axis, the rate of convergence of the algorithm and the quality of the final solution decrease for smaller angular displacements between frames. In the limit of the continuous formulation, the solution is no longer stable. The perspective formulation behaves similarly; in this case, there is also a dependence of convergence rate and quality of the solution on the size of the spatial displacements between frames for objects translating through space.

3. MIDDLE VISION AND INTEGRATION

Biological vision systems achieve their high degree of efficiency, robustness and reliability at recognition and navigation in highly variable environments through the integration of many visual sources. The simple task of locating object boundaries can be performed far more effectively by integrating evidence of discontinuities in image intensity, stereo disparity, speed and direction of motion, and texture than by using evidence from a single visual source on its own. We are now attacking the general integration problem, with the goal of deriving an accurate, robust and explicit representation of the structure of the environment and its surface properties, through the fusion of a variety of early vision algorithms. Our work during the last few years has provided a characterization of *early vision* and developed a unified approach to many early vision problems. We have now efficient algorithms that solve specific problems such as edge detection, stereo, shape from shading, motion measurement and interpretation, and so on. Though more work needs to be done in these specific areas, it is now time to address the integration of multiple visual sources into a robust vision system. We refer to this stage of visual processing as *middle vision*.

The problem of middle vision is to integrate information about the individual physical processes underlying image formation to form a unified and robust description. The standard regularization framework that we have developed for early vision by itself does not provide a satisfactory way of integrating the individual processes to compute rich surface descriptions, which we call $2\frac{1}{2}$-D sketches, a term introduced by Marr. The reason is that different processes can be combined by standard regularization only in a linear way (the cost functional is quadratic, originating linear Euler-Lagrange equations). Linear averaging of different processes — say depth information from stereo and from motion — is not a flexible enough integration method. Even more importantly, no instances of standard regularization can handle discontinuities, because the solution space is restricted to generalized splines (Poggio et al., 1985; Bertero et al., 1986). As we will explain later, we believe that detecting and representing discontinuities (for instance depth discontinuities) is a key part of the integration step (see Poggio, 1985).

Fortunately, the same regularization ideas that are useful in dealing with early vision provide a starting point for treating middle vision integration problems as well. We have developed a powerful extension to regularization that promises to deal simultaneously with discontinuities and with vision module integration. This extension is based on several new ideas and on the use of coupled Markov Random Fields, introduced recently by Geman and Geman at Brown University and extended by Jose Marroquin, Tomaso Poggio and Sanjoy Mitter.

This new approach contains the standard regularization method for vision as a special case.

3.1. A Regularization Approach to the Integration Problem: MRFs

A Markov Random Field (MRF) on a lattice can be represented as a lattice of sites, each one with a random variable. The value depends probabilistically on the value of neighboring sites. The rules governing this local dependence can be given in a variety of ways and can be made to capture constraints such as continuity of a surface (if the MRF represents depth values). Several MRFs, each associated with a different visual source, can be coupled together in complex ways. Furthermore, discontinuities can also be described by appropriate MRFs capturing the continuity of surfaces between boundaries and the discontinuity of surfaces at boundaries.

In more detail, a MRF is the two-dimensional extension of Markov chains. It has the property that the probability distribution of the configurations of the field can always be expressed in the form of a Gibbs distribution:

$$P_f(f) = \frac{1}{Z} e^{-\frac{1}{T_0}U(f)}$$

where Z is a normalizing constant, T_0 is a parameter (known as the "natural temperature" of the field) and the "Energy function" $U(f)$ is of the form:

$$U(f) = \sum_C V_C(f)$$

where C ranges over the "cliques" associated with the neighborhood system of the field, and the potentials $V_C(f)$ are functions supported on them (a clique is either a single site, or a set of sites such that any two sites belonging to it are neighbors of each other).

As an example, the behavior of piecewise constant functions was modeled by Geman and Geman using first order MRF models on a finite lattice L with generalized Ising potentials:

$$V_C(f_i, f_j) = \begin{cases} -1, & \text{if } |i-j| = 1 \text{ and } f_i = f_j \\ 1, & \text{if } |i-j| = 1 \text{ and } f_i \neq f_j \\ 0, & \text{otherwise.} \end{cases}$$

$$f_i \in Q_i = \{q_1, \ldots, q_M\} \quad \text{for all } i \in L$$

We use a free boundary model, so that the neighborhood size for a given site will be: 4, if it is in the interior of the lattice; 3, if it lies at a boundary, but not at a corner, and 2 for the corners.

The Gibbs distribution:

$$P_f(f) = \frac{1}{Z}\exp[-\frac{1}{T_0}U_0(f)]$$

$$U_0(f) = \sum_{i,j} V(f_i, f_j) \qquad (2$$

defines a one parameter family of models (indexed by T_0) describing piecewise constant patterns with varying degrees of granularity.

We assume that the available observations g are obtained from a typical realization f of the field by a degrading operation (such as sampling) followed by corruption with noise (the form of whose distribution is known), so that the conditional distribution can be written as:

$$P_{g|f}(g; f) = \exp -\alpha\hat{\epsilon} \qquad (3)$$

with $\hat{\epsilon} = \sum_{i \in S} \Phi_i(f, g_i)$, where $\{\Phi_i\}$ are some known functions, and α is a parameter.

The posterior distribution is obtained from Bayes rule:

$$P_{f|g}(f; g) = \frac{1}{Z_P}\exp[-U_P(f; g)] \qquad (4)$$

with

$$U_P(f; g) = \frac{1}{T_0}U_0(f) + \sum_{i \in S}\Phi(f, g_i) \qquad (5)$$

For example, in the case of binary fields ($M = 2$) with the observations taken as the output of a binary symmetric channel (BSC) with error rate ϵ, we have:

$$P(g_i|f_i) = \begin{cases} (1-\epsilon), & \text{for } g_i = f_i \\ \epsilon, & \text{for } g_i \neq f_i \end{cases}$$

The posterior energy reduces to:

$$U_P(f; g) = \frac{1}{T_0}\sum_{i,j} V(f_i, f_j) + \alpha\sum_i (1 - \delta(f_i - g_i)) \quad (6)$$

where $f_i \in \{q_1, q_2\}$;

$$\delta(a) = \begin{cases} 1, & \text{if } a = 0 \\ 0, & \text{otherwise.} \end{cases}$$

and

$$\alpha = \ln\left(\frac{1-\epsilon}{\epsilon}\right) \qquad (8)$$

Appropriate optimal estimators based on the *a posteriori* distributions can be derived in terms of the marginals and the mean of the posterior distribution. The main obstacle for the practical application of these results lies in the formidable computational cost associated with the exact computation of marginals and means even for lattices of moderate size. We have, however, derived a series of algorithms closely related to the Metropolis method and to *annealing* techniques that will permit us to approximate these quantities.

3.1.1. The Central Role of Discontinuities

One of the most important constraints for recovering surface properties is that the physical processes underlying image formation are typically smooth: surfaces are mostly continuous in depth and orientation and so with reflectance and illumination. The smoothness property is captured well by standard regularization and exploited in its algorithms. Surfaces and their properties, however, are not always smooth: they are smooth *almost* everywhere, but not at discontinuities. Lines of discontinuity are usually continuous, relatively smooth, nonintersecting curves. It is critical to detect reliably the discontinuities since they usually represent the most important locations in a scene: depth discontinuities, for instance, correspond often to the boundaries of an object or of a part. Furthermore, discontinuities play a critical role for fusing information from different physical processes. The reason is clear: in smooth regions the physical processes are coupled together by the imaging equation — all contribute to image formation. The exact coupling is however difficult to know precisely — it depends on quantities such as the form of the reflectance function. The effects of discontinuities is instead very robust and qualitative: for instance depth discontinuities usually correspond to intensity edges. Therefore, discontinuities are ideal places for integrating information (see Poggio, 1985 for more details on these ideas).

Often partial information about discontinuities in a single process can be detected relatively easily. Several types of motion discontinuities can be measured with simple operations on the time dependent intensity array, especially if the interframe interval is very small. Albedo discontinuities also may often be detectable in terms of simple operations. Intensity edges are detected quite reliably by the Canny edge detector, that we have now implemented on the Connection machine system. The fast, rough detection of discontinuities performed by these early operations is however incomplete: it must be refined by integrating them across processes and by exploiting the constraints of continuity of discontinuities.

3.1.2. Using Coupled MRFs to Integrate Vision Modules

The idea suggested by Poggio (1985) is to associate a MRF on a lattice to each physical process to be integrated and another (binary) MRF to its discontinuities. The lattices are coupled to each other to reflect the interdependence of the corresponding processes in image formation. Thus the various MRFs mirror the different physical events that underlie image formation: surface, surface discontinuities, spectral albedo, albedo discontinuities, shadows, surface normal, and so on.

Some of the processes underlying image formation (under oversimplified assumptions) are

1) the surface depth $z = f(x, y)$

2) the albedo $\rho(x, y, \lambda)$

3) the *effective irradiance* that usually depends on the surface normal and on the illuminations

4) the surface normal $\vec{n}$

5) the illumination $I(x, y, \lambda)$

6) Other minor processes such as specularities or visible light sources that may locally veto or switch off some of the previous processes.

7) Motion, for the time dependent case.

Physical constraints apply to each of these processes independently. In addition, there are constraints between these processes (for instance between z and $\vec{n}$). The image data constrain the way the processes combine. Note that consideration of time dependence will introduce additional powerful constraints such as rigidity. The constraints on z, $\vec{n}$, ρ are local conditions (such as smoothness, necessary mainly because of its regularizing role in the face of omnipresent noise) valid everywhere *except* at discontinuities. As we discussed earlier, discontinuities are critically important and detected very early.

The local potentials underlying the *a priori* probability distribution of the MRFs represent the constraints on the physical processes (smoothness, positivity, bounds, etc.); the coupling between MRFs represents the compatibility constraints *between* processes. The machinery of coupled MRFs provides an ideal tool to impose the local constraints of smoothness (plus others, such as positivity, values within certain bounds, etc.) allowing at the same time an explicit role for discontinuities through the *line process* of Geman and Geman (and similar processes such as *occlusions*). The new and powerful idea here is to incorporate additional *observable* discontinuity processes, which are algorithms specialized to detect sharp changes in the observed properties of motion, stereo disparity, texture, and so on. The observable discontinuities provide a fast and coarse solution to the segmentation problem. Using the MRFs for estimating the fields gives an increasingly more precise solution, simultaneously *filling in* the continuous regions that are only sparsely observable from the specialized algorithms for discontinuity detection. The solution at each iteration is available to later modules, such as recognition.

3.1.3. An Example: Integrating Stereo Data and Intensity Edges

For recognition it is important to extract reliable object boundaries. Even for systems such as the *RAF* recognition system (see later), which require only sparse sensory data, it is useful to obtain this data from object boundaries in the scene. Edges in intensity are in general not fully sufficient by themselves as indicators of object boundaries, because they are typically incomplete and confound different physical cues such as shadows, highlights, surface orientations and surface depth. Stereo data, on the other hand, as given by all algorithms developed so far is typically plagued by noise exactly at discontinuities in depth. It seems that intensity edges could be usefully combined with sparse depth data to recover more reliable depth discontinuities. At the same time, one can begin to distinguish among the intensity edges those that are due to depth discontinuities from those due to other physical causes. This is again important for later modules.

Ed Gamble and T. Poggio are presently applying the MRF based algorithms (at present running on the Connection machine system) to stereo data obtained from natural images, and use at the same time Canny's intensity edges. In this way they are able to find a better map of the depth discontinuities and classify in part the type of intensity edges. In addition to its important applications for photo interpretation and recognition, this problem offers an opportunity to study and solve several basic problems in the integration of information with MRFs:

- *Tolerance* to imperfect alignment of the different maps.
- *Multiresolution representations.*
- *Parameter estimation.*

3.2. Integrating Shading and Stereo

Wildes has considered how stereo and shading can aid one another in the recovery of 3D surface shape. In particular: (1) Stereo and binocular brighteness measurements together set boundary conditions for shape from shading (s.f.s.); whereas (2) S.f.s information can interpolate between sparse stereo data; and (3) S.f.s and binocular brighteness measurements can be used to correct incorrect stereo matches resulting from self shadows and high lights. To utilize s.f.s information, it is necessary to recover reflectance parameters. These are estimated by looking at the extrema in the derivatives of an assumed parametric form of the reflectance function. An implementation recovering both 3D surface shape and surface reflectance parameters has been tested on a wide range of synthetic images with generally positive results. Although this implementation specifically sought to link Marr-Poggio-Grimson stereo with Horn et. al. s.f.s., the general method should be applicable to linking any feature point stereo system with any defect correction s.f.s. system. More details are given in the paper in these Proceedings.

4. OBJECT RECOGNITION

4.1. Recognition from Matched Dimensionalities

In earlier reports, we have described the work of Grimson and Lozano-Perez on object recognition from sparse, noisy sensory data. This work developed a technique (called RAF) for recognizing occluded objects, for which we have polyhedral models, from simple measurements of the position and surface orientation of small patches of surface. The technique searchs for consistent matchings between the faces of the object models and the sensory measurements, using simple but powerful geometric constraints in a standard backtracking tree search. The technique is characterized by requiring a matching of the dimensionality of the problem and the data. That is, if we are dealing with flat objects on a stable plane, so that the objects undergo only two degrees of translational freedom, and one degree of rotational freedom, then we only require two dimensional sensory input, such as the positions and orientations of edges in an image. If we are dealing with solid objects in arbitrary positions, so that the objects undergo three degrees of translational freedom, and three degrees of rotational freedom, then we require three dimensional sensory input, such as the positions and orientations of surfaces from range data.

In the past year, we have considered a number of problems in recognition associated with this approach. First, we have directly extended the RAF system in several ways. We have begun to address the problem of parameterized objects, by extending the geometric shapes to allow for object variation. We have considered three classes of parameterized objects. The first class is objects that scale in size, the second class is objects with rotational degrees of freedom, such as a pair of scissors, and the third class is objects that stretch along an axis. In each case, one can derive expressions for the geometric relationship between two edges as a function of the free parameters. Given such a model of an object, when interpreting sensory data, we simply modify the tree search to pass the range of feasible values for the free parameters as the tree of interpretations is explored. This allows one both to determine the position and orientation of the object, and the range of feasible values for the rotational or stretching degree of freedom. Second, we have extended the RAF method to deal with libraries of subparts. Previously, we had considered configuration hashing techniques as a means of reducing the search associated with recognition. Such hashing schemes can be straightforwardly extended to deal with multiple objects, and we have tested this scheme on the problem of identifying multiple instances of multiple types of objects from noisy data.

We have also considered other related approaches to recognition, spurred in part by properties of the RAF technique. David Clemens has investigated the tradeoffs between hypothesis-driven and data-driven recognition methods. He has investigated the use and representation of features for guiding efficient search, as well as a number of hashing schemes and matching strategies for controlling the inherent complexity of the search process.

Gil Ettinger has developed a model-based recognition system that integrates a curvature primal sketch representation (based on earlier work by Brady and colleagues) with a constrained search recognition method (based on the RAF system of Grimson and Lozano-Perez). The system benefits from the rich vocabulary of the representation and the spatial constriants of the object features in order to achieve the goals of robustness, efficiency, and extensibility. The technique takes advantage of the scale-space nature of the curvature primal sketch representation to deal with classes of objects. In particular, the system addresses the issue of automatic determination of object subparts and the recognition of objects from a model library, in which the subparts are allowed to vary, by allowing free parameters to range over a set of values.

David Jacobs is developing a technique for automatically segmenting sensory data into groups that are likely to come from a single object. The segmentation scheme is based on general principles concerning the types of objects expected to have produced the image. This technique is crucial to efficient recognition, since it provides a powerful means of restricting the portions of the search space that must be explored by a recognition engine. It also enables efficient indexing into libraries of objects. A prototype system has been built and successfully tested on libraries of simple objects. Extensions to complex objects, and extensions to three-dimensional objects are currently underway.

We have also considered parallel implementations, on the Connection Machine, of RAF based recognition schemes. John Harris and Anita Flynn developed two preliminary designs for Connection Machine implementations of the original Grimson and Lozano-Perez methods. Recently, Todd Cass has derived a new parallel recognition method, based on the same principles of the RAF system, which is currently being implemented and tested on the Connection Machine.

4.2. Another Approach to Object Recognition: From 2D to 3D

In these Proceedings Huttenlocker and Ullman present a new approach to recognition, where an object is first *aligned* with an image using a small number of pairs of model and image features, and then the aligned model is compared directly against the image. For instance, the position and orientation of a rigid object in three-space can be determined from three pairs of corresponding model and image features. Given a set of possible alignments, the best match of a model with an image is given by the alignment which maps the largest number of model features onto image features. In contrast, existing recognition systems find the best match by searching for the largest set of model and image feature pairs which are consistent with a single position and orientation of a rigid object. Sincethisspace of possibilities is exponential, various techniques are used to limit the search. Thus by using a small fixed number of features to determine position and orientation, the alignment process avoids structuring the recognition problem as an exponential search.

To demonstrate this recognition method, they tested examples of recognizing flat objects with arbitrary three-dimensional position and orientation. The recognition system chooses features for alignment using a hierarchical edge-based shape description. Coarse scale features are labeled using both the shape of the feature and the structure of the hierarchy at the next finer level. This produces richly descriptive features for use in alignment. Fine scale features are then used to determine how well a given alignment matches the model with the image.

4.3. Matching and Indexing in Parallel

As described in these Proceedings, W. Lim has implemented an algorithm in symbolic high level vision on the Connection machine. It takes a library of qualitative surface models of rocks or mountains and loads them all into the Connection machine. Candidate images are processed to extract object boundaries which are then matched in parallel to all library models.

5. THE VISION MACHINE

As an integration testbed of much of the vision effort outlined in this report we are beginning to use a system based on a Symbolic 3640 connected to the Connection machine. The MIT two-camera stereo system (the equivalent of an eye-head system) described in the last Proceedings, is the input device of the machine. Work in the last two months by Little, Cass, Villalba, Buelthoff, Gamble, Drumheller has made it possible to use the system to "look around", grab images, compute Canny's edges, estimate optical flow, derive stereo depth and integrate intensity data with depth in close to real time. Over the next few months we plan to implement other early vision modules on the system and integrate them.

5.1. The Connection Machine System

This section gives a brief description of the Connection Machine^{TM}, the parallel computer which is the main computational engine of the Vision machine.

The Connection Machine^{TM} (CM) is a fine-grained parallel supercomputer originally conceived at the AI Lab and then developed and produced by Thinking Machines Corporation for research in artificial intelligence. The full prototype contains 65,536 1-bit serial processors capable of communicating with each other by means of two distinct mechanisms (the version at the AI Lab has 16,434 1-bit processors). One mechanism has the topology of a *boolean 16-cube* and is called the *router network*, or simply "the router." The other mechanishm is a two-dimensional four-connected x-y grid called the *north-east-west-south* connections, or "NEWS." Short-range communication between processors is very efficient using NEWS. Long-range communication is very efficient using the router. Therefore, NEWS is used for operations requiring fast *local* communication, such as convolutions and relaxation algorithms. The router is used for *global* operations such as permutation, sorting, merging, summing, histogramming, region-growing, and image sampling. Each processor has about 4K bits of memory available to the user. The CM is programmed from a host computer, in our case a Lisp Machine, which broadcasts the same instructions to every processor. The operations performed by a particular processor depend on the data contained in that processor's memory. See (Hillis, 1985) for a more detailed description of the Connection Machine.^{TM}

5.2. Parallel Algorithms

In a strictly related effort we are developing parallel versions of our algorithms on the Connection machine and testing their performance. Our work amounts to developing a new model of computation, almost a different way of thinking how to solve various kinds of problems in vision. A paper by J. Little in these Proceedings give details of some of our work in this area. We are presently implementing a range of vision utilities: convolution, edge detection (T. Cass), optical flow (Little and Buelthoff), motion discontinuities (Spoerri; Little and Buelthoff), stereo (Drumheller and Poggio), MRF-based integration (Gamble and Poggio), some visual routines (Mahoney), are all running on our Connection machine.

6. NAVIGATION

6.1. Experiments in Real-Time Direct Passive Visual Navigation of a Mobile Robot

We have begun to experiment with some of the ideas developed by Horn, Negahdaripour and others for using optical flow from passively acquired images for mobile robot navigation. These ideas assume that the motion field of a scene corresponds to optical flow in an image, and that lighting conditions are such that as the robot moves individual surface patches reflect the same amount of light to it.

The general *image brightness change equation* relates the 6 degree of freedom motion of a camera to the changing brightness levels which will appear in images it is collecting. If the camera is moving with rotational component ω and translational component t then we can say that for every point in an image

$$E_t + v \cdot \omega + \frac{1}{Z} s \cdot t = 0$$

where Z is the distance to the place in the world corresponding to the image point, E_t is the temporal brightness gradient at the image point, and v and s are some moments of the spatial brightness gradients at the image point.

Now suppose that we know the motion of the robot is pure translation in the z direction, i.e. the robot has a forward looking camera and it is translating in the forward direction. We can simplify the above equation and compute the distance to a point in time units (i.e. as time to collision) with

$$-\frac{xE_x + yE_y}{E_t}$$

where E_x, E_y and E_t are the derivatives of image brightness in directions x and y, and time respectively. If this equation is accurate then we can use it to segment an image into obstacles which are in the robots path. It delivers range in units of time to collision, so it is naturally calibrated for the robot's current velocity.

Brooks, Flynn and Connell have run some experiments on sequences of images taken by the mobile robot moving towards obstacles like oscilloscopes and chairs, computing the above quantity at every point. The images were 32 × 32 pixels; sufficient resolution for local obstacle avoidance. The results come out looking like noise. The assumptions made in the first paragraph do not hold at many points in real images. However, there is a source of additional constraint which will let us prune out most of those noisy values.

The above quantity is actually a truncated Taylor series expansion of

$$-r\frac{E_r}{E_t}$$

where r is the distance from the center of expansion. The quantity

$$-E_t/E_r$$

is the velocity of optical flow way from the focus of expansion. This must always be positive, and its x and y components must both be away from the FOE. This implies that each of xE_x and yE_y should have opposite signs to E_t. We filter out points in the image for which this is not true. We further throw away any points which suggest a collision sooner than the proximity sensors mounted on the robot suggest. The result is a sparse qualitatively accurate depth map, suitable for input to local obstacle avoidance algorithms. The total computation takes less then 10000 arithmetic operations.

The computation is extremely sensitive to noise in E_t and we need to improve our estimation technique for this quantity (so far we have simply used brightness difference between successive images). The computations are also sensitive to the image coordinates of the assumed focus of expansion. We need to incorporate a calibration ntechnique which updates the estimate of the FOE. Ideally we will find a technique that is computationally cheap enough to be used on every series of images in real time.

REFERENCES

Bertero, M., T. Poggio and V. Torre. "Ill-Posed Problems in Early Vision," Massachusetts Institute of Technology Artificial Intelligence Laboratory Memo 924, 1986.

Brooks, M.J. and Berthold K.P. Horn. "Shape and Source from Shading," Massachusetts Institute of Technology Artificial Intelligence Laboratory Memo 720, 1985.

Canny, John F. "Finding Edges and Lines," Massachusetts Institute of Technology Technical Report 720, 1983.

Cornog, Katherine H. "Smooth Pursuit and Fixation for Robot Vision," Massachusetts Institute of Technology Department of Electrical Engineering and Computer Science Master's Thesis, 1985.

Drumheller, M. and T. Poggio. "On parallel stereo," in: **Proc. IEEE Int. Conf. on Robotics and Automation**, 1986.

Flynn, Anita M. "Redundant Sensors for Mobile Robot Navigation," Massachusetts Institute of Technology Department of Electrical Engineering and Computer Science Master's Thesis, 1985.

Geman, Stuart and Don Geman. "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, **6**, 1984.

Grimson, W.E.L. **From Images to Surfaces**, Massachusetts Institute of Technology Press, Cambridge, Mass., 1981.

Grimson, W.E.L. "A computational theory of visual surface interpolation," *Phil. Trans. R. Soc. London B*, 1982.

Grimson, W.E.L. "Surface consistency constraints in vision," *Computer Vision, Graphics, and Image Processing*, **24**, 1983.

Grimson, W.E.L. "The Combinatorics of Local Constraints in Model-Based Recognition and Localization from Sparse Data," Massachusetts Institute of Technology Artificial Intelligence Laboratory Memo 763, 1984.

Grimson, W.E.L. "Sensing Strategies for Disambiguating Among Multiple Objects in Known Poses," Massachusetts Institute of Technology Artificial Intelligence Laboratory Memo 855, 1985.

Grimson, W.E.L. and T. Lozano-Pérez. "Model-Based Recognition and Localization from Sparse Range or Tactile Data," *International Journal of Robotics Research*, 3 1984.

Grimson, W.E.L. and T. Lozano-Pérez. "Model-based Recognition and Localization from Tactile Data," IEEE Computer Society Int. Conf. on Robotics, Atlanta, March 1984.

Grimson, W.E.L. and T. Lozano-Pérez. "Recognition and Localization of Overlapping Parts from Sparse Data in Two and Three Dimensions," IEEE Computer Society Int. Conf. on Robotics, St. Louis, March 1985.

Grimson, W.E.L. and T. Lozano-Pérez. "Model-Based Recognition and Localization From Sparse Range Data," in: **Techniques for 3-D Machine Perception**, A. Rosenfeld (ed), North-Holland, Amsterdam, 1985.

Grimson, W.E.L. and T. Lozano-Pérez. "Search and Sensing Strategies for Recognition and Localization of Two and Three Dimensional Objects," Third Int. Symp. on Robotics Research, Gouvieux, France, October 1985. Published by MIT Press, Cambridge, Mass.

Grimson, W.E.L. and T. Lozano-Pérez. "Recognition and Localization of Overlapping Parts from Sparse Data," in **Three-Dimensional Vision Systems**, T. Kanade (ed), Kluwer Academic Publishers, 1985.

Grimson, W.E.L. and T. Lozano-Pérez. "Recognition and Localization of Overlapping Parts from Sparse Data," Massachusetts Institute of Technology Artificial Intelligence Laboratory Memo 841, 1985.

Hildreth, E.C. "Edge Detection," Massachusetts Institute of Technology Artificial Intelligence Laboratory Memo 858, 1985

Hildreth, E.C. and N.M. Grzywacz, "The incremental recovery of structure from motion: position versus velocity-based formulations," *Proc. IEEE Workshop on Motion: Representation and Analysis*, Charleston, SC, 1986, pp. 137-143. Also to appear in *J. Opt. Soc. Amer. A*, 1987.

Hillis, D. "The Connection Machine," Massachusetts Institute of Technology Department of Electrical Engineering and Computer Science Ph.D. Thesis, 1985.

Horn, B.K.P. "Obtaining Shape from Shading Information," in: *The Psychology of Computer Vision*, P.H. Winston (ed), McGraw-Hill, New York, pp. 115-155, 1975.

Horn, B.K.P. "Understanding image intensities," *Artificial Intelligence*, 8, 201–231, 1977.

Horn, Berthold K.P. **Robot Vision**, Massachusetts Institute of Technology Press, Cambridge and McGraw-Hill, New York, 1985.

Horn, B.K.P. and B.G. Schunck. "Determining optical flow," *Artificial Intelligence*, **17**, 185-203, 1981.

Hurlbert, A. and T. Poggio. "Do computers need attention?" *Nature*, /bf 321, 12, 1986.

Kirkpatrick, S., C.D. Gelatt, Jr. and M.P. Vecchi. "Optimization by simulated annealing," *Science*, **220**, 1983.

Lozano-Pérez, T. and W.E.L. Grimson, "Recognition and localization of overlapping parts from sparse data," Second Int. Symp. on Robotics Research, Kyoto, Japan, August 1984. Published by MIT Press, Cambridge, Mass.

Marr, David. "Early processing of visual information," *Phil. Trans. R. Soc. London B* **275**, 1976.

Marr, David and Keith Nishihara. "Representation and recognition of the spatial organisation of three dimensional shapes," *Proc. R. Soc. Lond. B.* **200**, 1978.

Marr, D. and E. Hildreth. "Theory of edge detection," *Proc. R. Soc. London*,B **207**, 1980.

Marr, D. and T. Poggio. "A cooperative stereo algorithm," *Science* **194**, (1976).

Marroquin, J.L. "Design of Cooperative Networks," Massachusetts Institute of Technology Artificial Intelligence Laboratory Working Paper 253, (1983).

Marroquin, J. "Surface Reconstruction Preserving Discontinuities," Massachusetts Institute of Technology Artificial Intelligence Laboratory Memo 792 (1984).

Marroquin, J. "Probabilistic Solution of Inverse Problems," Ph.D. Thesis, Massachusetts Institute of Technology (1985).

Marroquin, J. "Surface Reconstruction Preserving Discontinuities," Massachusetts Institute of Technology Artificial Intelligence Laboratory Memo 792, 1984.

Marroquin, J. "Optimal Bayesian Estimators for Image Segmentation and Surface Reconstruction," Massachusetts Institute of Technology Artificial Intelligence Laboratory Memo 839, 1985.

Marroquin, J., S. Mitter and T. Poggio. "Probabilistic solution of ill-posed problems in computational vision.," in: **Proceedings of the Image Understanding Workshop**, L. Baumann (ed), Scientific Applications International Corporation, 1985.

Mayhew, J.E.W. and J.P. Frisby. "Psychophysical and computational studies towards a theory of human stereopsis," *Artificial·Intelligenec* **17**, (1981).

Metropolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller. "Equation of state calculations by fast computing machines," *J. Phys. Chem.* **21**, 1953.

Poggio, Tomaso. "Early vision: from computational structure to algorithms and parallel hardware," *Computer Vision, Graphics, and Image Processing*, **31**, 1985.

Poggio, T. "Integrating Vision Modules with Coupled MRF's," Massachusetts Institute of Technology Artificial Intelligence Laboratory Working Paper 285, 1985.

Pollard, S.B., J.E.W. Mayhew and J.P. Frisby. "Disparity gradients and stereo correspondences," *Perception*, in press.

Prazdny, K. "Detection of binocular disparities," *Biological Cyberenctics* **52** (1985).

Reichardt, W., T. Poggio and K. Hausen. "Figure-ground discrimination by relative movement in the visual system of the fly. Part II: Towards the neural circuitry," *Biol. Cyber.* **46** pp. 1-30, 1983.

Poggio, Tomaso, Vincent Torre and Christof Koch. "Computational vision and regularization theory," *Naturc* **317**, 1985.

Poggio, T. and A. Verri. "Regularization Theory and Shape Constraint," Massachusetts Institute of Technology Artificial Intelligence Laboratory Memo 916, 1986.

Poggio, Tomaso, Harry Voorhees and Alan L. Yuille. "Regularizing Edge Detection," Massachusetts Institute of Technology Artificial Intelligence Laboratory Memo 776, 1984.

Tikhonov, A.N. and V.Y. Arsenin. **Solution of Ill-Posed Problems**, Winston and Wiley Publishers, Washington D.C., 1977.

Torre, Vincent, and Tomaso Poggio. "On Edge Detection," Massachusetts Institute of Technology Artificial Intelligence Laboratory Memo 768, 1984.

Ullman, Shimon. "Visual routines," *Cognition,* 18, 1984.

Ullman, S. "The Optical Flow of Planar Surfaces," Massachusetts Institute of Technology Artificial Intelligence Laboratory Memo 870, 1985.

Ullman, S. "Maximizing rigidity: the incremental recovery of 3–D structure from rigid and rubbery motion," *Perception* **13**, pp. 255-274, 1984.

Yuille, A.L. and T. Poggio. "A generalized ordering constraint for stereo correspondence," Massachusetts Institute of Technology Artificial Intelligence Laboratory Memo 777 (1984).

Verri, A. and T. Poggio. "Motion Field and Optical Flow: Qualitative Properties," Massachusetts Institute of Technology Artificial Intelligence Laboratory Memo 917, 1986.

AUTHOR

Tomaso Poggio is with the Artificial Intelligence Laboratory; Massachusetts Institute of Technology, NE43-787; 545 Technology Square; Cambridge, MA 02139

Summary of Progress in Image Understanding at the University of Massachusetts

Allen R. Hanson and Edward M. Riseman

Computer and Information Science Department
University of Massachusetts at Amherst

ABSTRACT

Image Understanding research at the University of Massachusetts encompasses a range of research, most of which is directed towards the integation of a diverse set of processes to achieve a general real-time knowledge-based interpretation system. In particular we are concentrating on integrating projects involving object identification in static images, depth recovery from motion analysis, a real-time parallel architecture, and mobile vehicle navigation. This system will be applied to a variety of task domains of natural scenes including road scenes and aerial images, and will also be used to control a mobile robot moving through both known and unknown outdoor domains.

This summary documents several areas of research at the University of Massachusetts that are entirely or partially supported under the DARPA image understanding program. The work, much of which is documented in papers in these proceedings, is divided into several areas:

1. Knowledge-Based Vision
2. Perceptual Organization (intermediate processing)
3. 3D Models, Matching, and Surface Recovery
4. Mobile Robot Navigation
5. Image Understanding Architecture
6. Motion Analysis
7. Low-Level Vision

1 Knowledge-Based Vision

A central problem in image understanding is the representation and use of all available sources of domain knowledge during the interpretation process. Each of the many different kinds of knowledge that may be relevant during the interpretation process imposes different kinds of constraints on the underlying representation and may lead to very different kinds of strategies for its effective use. Over the past several years, we have developed the notion of a 'schema' as the basic unit of knowledge representation in the VISIONS system. Within the schema system image interpretation is the process of instantiating a subset of schemas to build a description of the three-dimensional scene which gave rise to the image. Knowledge is represented in an abstraction hierarchy of schema nodes by part/subpart descriptions, class/subclass descriptions, and expected relationships between schemas; the resultant hierarchical graph constitutes the VISIONS knowledge network. This work has evolved for a long period of time, with recent work documented in [19,23,45].

Each schema node may be viewed as a 'packet' of information related to the object being described, including properties and relations of extracted image events as well as control information expressed in the form of interpretation strategies. One or more of these strategies are executed when a schema is instantiated (i.e. when a copy is activated), to process a specific area of the image. Schema activation may be either bottom-up, where image descriptions imply the potential relevance of a schema, or top-down as the result of the context of a partial interpretation written on a blackbord communication structure by other schemas. Many schemas may be active at any one time, and the interpretation strategies provide control over the parallel interpretation processes and make use of a set of object-independent processes called knowledge sources. In our system schemas communicate indirectly by posting object hypotheses on the blackboard.

The system is organized around three levels of data representation and types of processing. At the low-level, the representations are in the form of numerical arrays of sensory data with processes for extracting the image events that will form the intermediate representation. At the intermediate level, the representation is composed of symbolic tokens representing regions, lines, surfaces and the attributes of these primitive elements (which might include local motion and depth information). The intermediate representation is stored in a data base called the intermediate symbolic representation (ISR) which supports grouping (perceptual organization) and information fusion processes that are employed to develop aggregations of existing tokens to form new tokens. At the high level, the representation is a set of object hypotheses and active schema instances which control the intermediate and low-level processes. Control initially proceeds in a data-directed manner and later is significantly top-down in a knowledge-directed manner.

Based on our experience with an initial implementation of the schema system and a set of experiments designed to interpret reasonably complex house scenes [23,24,45], a new schema system and support environment has been designed and partially implemented [19]. Two new tools, the Intermediate Symbolic Representation and the Schema Shell, have been developed and are currently being tested and extended using the

This work was supported by DARPA under Contracts N00014-82-K-0464, DACA76-85-C-0008, and DACA76-86-C-0015, by AFOSR under contract F49620-86-C-0041 and 86-0021, by DMA under contract 800-85-C-0012 and by the National Science Foundation under grants DCR-8318776, and DCR-8500332.

interpretation of road scenes as a second experimental task domain. A third experimental domain of aerial image analysis for cartography applications is planned for the near future.

The input to high-level vision processes is *intermediate-level* data, which is the output from low-level processes such as line extraction and region segmentation, and of intermediate-level processes of grouping and selection. In our environment, intermediate level image descriptions are stored in the Intermediate Symbolic Representation, (or ISR). The ISR is a database which has been custom-built for the efficient storage, manipulation, and retrieval of abstract image data. The fundamental unit of representation is the *token*, each of which has a unique name, and a list of feature slots. The ISR can be used to store anything that can be fully characterized by a list of features and values; some of the image events currently stored include region segments, extracted edge lines, areas of homogenous texture, rectilinear line groups, and region-line relations. The benefits which result from imposing a uniform representation and user interface on all intermediate level tokens are enormous. It now becomes natural to think in terms of multistage and hierarchical grouping processes which take in tokens at one level of abstraction and produce tokens at the next higher level [39]. It also becomes more tractable to compare different types of tokens, which is necessary, for example, when relating edge lines to the regions they intersect [6]. Of course, the sharing of results and the elimination of data reformatting routines are obvious advantages.

At the highest level, tokens are partitioned according to the *image* they were extracted from. There is also an intermediate level of partitioning called the *tokenset*. Each feature associated with the tokens in a tokenset has a name, a value slot, a data type, and a computation function. Since most tokens have a physical realization in an image, a special bitplane data type is provided for representing the subset of image pixels which are associated with a token. Operators exist for taking the intersection, union, and difference of token bitplanes.

The ISR supports efficient access functions to tokens and sets of tokens; some of these access functions are associative in nature in that tokens may be accessed by means of constraints on feature values. In addition, the features may be precomputed or computed on a demand basis when the token/feature slot is accessed. The ISR allows a schema to create a token during an interpretation, create its bitplane either from scratch or as some combination of token bitplanes, and access its features, at which time new feature values will be automatically calculated for the new token. Thus, it is possible for schemas to dynamically "correct" misleading segmentations based on combinations of top-down knowledge.

The Schema Shell is an environment tool that supports the development of large systems of schemas. Each schema contains knowledge about recognizing a class of objects. It has data declarations for collecting relevant information, and procedures for determining whether, when and how to ascertain that information. Parameterized instances of schemas are then invoked to interpret an image. The Schema Shell provides mechanisms for building schemas and simulats a distributed environment (until parallel hardware arrives) in which an arbitrary number of schema instances may run concurrently. Schema instances communicate through a central blackboard. At any point during its processing a schema strategy may write an arbitrary message to the blackboard. Every other schema is then free to read, erase or modify that message. This provides for a single, uniform communication mechanism between schemas which can also be easily implemented on a variety of distributed architectures.

Bottom-up activation of schemas can be accomplished by forming initial object hypotheses on the basis of attributes of the initial image description expressed in terms of lines and regions. Previously we have reported on rule-based approaches to initial hypothesis generation [6,24] which used a heuristic approach to forming constraints (rules) based on a theoretical Bayesian approach to maximum likelihood decisions over feature distributions. Recently, Lehrer and Reynolds [33] have extended the work and have developed a new object hypothesis system based on the Shafer-Dempster [17,40] theory of evidence. Their approach provides a more formal and theoretical foundation for the definition and interpretation of world knowledge. Object specific knowledge is defined automatically using statistical information obtained from a set of training object instances and a computationally efficient approach to the Dempster-Shafer theory of evidence is used for the representation and combination of evidence from multiple sources.

In this approach, the relationship between an object and its attributes is captured in a "plausibility" function. When applied to the primitive tokens (e.g. regions) the plausibility functions return evidence for or against an object hypothesis. The evidence from multiple plausibility functions is combined using an efficient computational algorithm to produce the final hypothesis. A large scale experiment is being planned for comparing and evaluating the results of the two systems.

2 Perceptual Organization (Intermediate Processing)

We are initially viewing the task of perceptual organization and grouping as the extraction of relevant structure from over-fragmented and incomplete descriptions and the construction of more abstract descriptions from less abstract ones. By this we mean algorithms which have as input the tokens produced by the low-level system and other grouping operations (region, lines, flow fields,...) and have as output more complex tokens generated by grouping strategies based on the *relations* between the tokens. The goal of this type of 'intermediate' level processing is the reducti of the substantial representational gap which exists betw he low level image descriptors and the primitives with which the high level semantic descriptions are constructed. The process of abstraction thus involves the search for events which can be more concisely described as a unit and which results in a description which may be more relevant to the evolving semantic interpretation.

Over the past two years some progress has been made in developing grouping algorithms at the intermediate level of representation. The intermediate symbolic representation, described briefly earlier, has been d veloped as the supporting representation for this work and several algorithms developed previously have been cast within this framework. A number of the local strategies for using the rank-ordered object hypotheses generated by the rule-structured initial object hypothesis system [24] can be viewed as grouping strategies. The extensions to this system developed by Belknap [5], which fuses information across multiple token types by means of relations expressed as rules, is also a form of grouping and has successfully generated object hypotheses from a combination of geometric and spectral features Boldt [11] has developed a scale-sensitive hierarchical algorithm for grouping collinear line segments into progressively longer segments on the basis of geometric properties of the hypothesized group as well as the similarity of image features along both sides of the component ines. A summary of these algorithms and a more comprehensive discussion of their relationship to perceptual organization and grouping may be found in [23].

As a result of these preliminary studies related to grouping, we [39,11] are developing a computational framework for geometric grouping and other organizational algorithms which addresses a set of overlapping issues. Clearly one must consider the extraction and representation of primitive tokens, the features of these tokens, and important relations between the tokens. In the case of geometric grouping algorithms this would include the extraction of lines and geometric relations such as co-linearity, parallelness, relative angle, and spatial proximity derived from the Gestalt Laws of perceptual organization. One must also provide the means for expressing domain constraints in terms of these relations; i.e. grouping *strategies* must be defined and invoked based on knowledge of the domain and the current state of the system. Finally the system must deal explicitly with the problem of search, and its relation to the objects in the domain which are to be hypothesised and identified. In general each step of any grouping strategy must apply constraints which either significantly reduce the search space and/or add important information to the descriptive power of the system.

A number of algorithms are being developed at UMASS which satisfy these requirements and a computational framework has been proposed for confronting the issues described above. We view the grouping and search processes as part of a four-stage iterative grouping and extraction strategy which can be summarized as follows:

- *Primitive Structure Generation*: These processes provide the primitives (regions, lines, possibly surfaces, and in general, tokens) which are the input to the grouping and hypothesis generation process described next.

- *Linked Structure Generation*: This step applies very general geometric constraints to obtain graphs within which search processes can be applied to identify specific objects of interest. For example rectilinear structures which would contain rectangles or other simple geometric structures. This is essential for generating search spaces of reasonable size.
- *Subgraph Extraction*: This step involves the extraction of specific structures "one step up" the abstraction hierarchy, and uses the linked structures to constrain the search.

- *Replacement and Iteration*: Having extracted more abstract tokens, these can now play the role of primitives in another pass of grouping and extraction.

In [11,12] this strategy has been applied with striking results for the purpose of extracting straight lines. In [39] this strategy is being applied for the purpose of extracting rectilinar structures. In unpublished work Lance Williams is developing an algorithm for using a flow field generated from a motion sequence of images, to assist in the straight line extraction and temporal grouping process with excellent preliminary results.

While many of the grouping algorithms discussed above are designed to be applied uniformly across an image, many of them are computationally intensive. In addition, it often does not make sense to apply them in a uniform fashion because they may not be applicable to all portions of the image. For example, the rectilinear grouping algorithm [39] probably should not be applied in heavily textured areas. Consequently, we are examining strategies in which the algorithms are applied selectively to those areas of the image for which they are most suited. Kohl [29] has been developing a schema-based system called GOLDIE for intelligently controlling the application of parametrized low- and intermediate-level processes on the basis of goals and constraints generated by the high-level interpretation system. Initially, GOLDIE (for Goal-Directed Intermediate Level Executive) was formulated as a goal-oriented resegmentation system which allowed top-down control over the low-klevel segmentatin processes and this remains an important kaspect of its function. However, it also has become clear that top-down control of the intermediate-level grouping processes is required; consequently GOLDIE has been extended to include these processes in its repertoire. Both the Boldt line grouping algorithm and a rule-based region merging algorithm are incorporated into it and we are examining further extensions. GOLDIE responds to requests from the interpretation processes through the goal structure by translating the goals into appropriate low- and intermediate-level process specifications and then executing the process. The constraints imposed on the output of the process can be quite general; if the resulting structure does not satisfy the request, the system attempts to generate other strategies, using whatever contextual and semantic knowledge is available, in order to meet the constraints.

3 3D Models, Matching and Surface Recovery

There have been two recent research efforts in our group directed towards 3D object recognition and surface recovery. Two-dimensional images provide us with cues to the three-dimensional structure of objects which can be used for recognition or description. We are exploring a methodology of generic (characteristic) views for model-based recognition. The primary feature which characterizes each of the generic views is the binary relationships between pairs of lines which are visible in the same view. For the problem of reconstructing surfaces we have adopted an approach of constructing the envelope of the object from changes of the contours under planar motion of the camera. What these two approaches have in common is that they both use geometric knowledge about contours. Our efforts are presented in abit more detail below.

Often small changes in the viewpoint will only produce small changes in the appearance of an object. If we measure the visibility of features, (e.g. whether or not an edge or vertex is visible), they will be stable over a wide range of viewpoints. Such a set of viewpoints of an object is called a generic view. The model of an object consists of all of its generic views. The classification of the types of features and transitions for smooth surfaces has been analyzed [15,26,28] and we have extended these results to piecewise smooth surfaces [16]. Piecewise smooth surfaces are made up of patches of smooth surfaces which meet at creases. This type of surface subsumes both polyhedra and smooth surfaces. If the pieces are planar, then the surface one gets is polyhedral. If there are no creases, then the surface is smooth.

Using this approach Kitchen and Burns are constructing a 3D modelbase of objects, which will be analysed in order to make predictions about the visual configurations that views of the objects will give rise to in an image. These predictions are being organized into a hierarchical structure with explicit sharing of predictions common to multiple views. At recognition time, extracted image features are to be matched against this hierarchy in order to quickly establish what view of what object is seen, even if there are many possible objects in the modelbase. Once the view is known and the correspondence determined between image features and 3D object parts, it is possible to solve numerically for the object's pose parameters, using general methods or view-specific methods where advantageous [25,36,37].

We are proceeding with an implementation and analysis of this approach as applied to recognizing rigid polyhedra. Currently a system for modelling polygonal prisms has been implemented, along with a graphics interface as a tool for exploring the geometry of predictions. More important, we have an initial system implemented for making predictions about the appearances of these prisms and organizing them into a hierarchy for recognition purposes. Work is also in progress on robust and efficient techniques for solving for object pose which are tailored for specific classes of views.

In a separate effort, Giblin and Weiss have mathematically analyzed the reconstruction of surfaces from profiles and have derived an algorithm to implement it. Information can be derived about the shape of an object from a single profile, and with multiple views the shape can often be determined uniquely. Based on this analysis they have found an algorithm which can be used to produce a depth map of the surface. However, for some applications it may not be necessary to produce a depth map at all (for example in recognition problems), and thus they have also provided an algorithm which computes Gauss and mean curvatures without first computing the depth map. It should be noted that the Gauss and mean curvatures have been used by other researchers to segment surfaces into patches which are convex, concave, hyperbolic, parabolic, or planar [7,13].

One of the basic problems to be solved is how to combine profiles from multiple views. In general, there is no way to identify a point on one profile with corresponding points on a profile from a different view, since for smooth surfaces they will not have any points in common. In fact, most stereo algorithms which are based on correspondence find the most similar point and assume it is the same. However, if the camera motion is known, then there is a method to identify points on two different profiles. In our work, the camera has been restricted to planar motion, so that planes parallel to the plane of motion induce a correspondence between the profiles. Nevertheless, it is possible for the profile to change qualitatively from one view to the next, and in order to understand this, the analogous problem for a curve in the plane has been analyzed. These view transitions create ambiguities in the reconstruction process. The criterion used to resolve this ambiguity is that the most likely solution is the one which minimizes the change in depth between adjacent views.

The mathematical approach to this problem is that a smooth surface without inflection points is the envelope of all of its tangent planes. However, there are two problems with this: how to compute the envelope of a family of planes and how to handle inflection points. With the assumption of planar camera motion, the envelope of planes problem has been reduced to that of computing the envelope of a family of lines in a plane, which Giblin and Weiss were able to solve. The algorithm has been applied experimentally to synthetic, noise-free data to reconstruct curves from their profiles with a high degree of accuracy. Future experiments for computing both a dense depth map and Gauss and mean curvatures will employ real data .

4 Mobile Robot Navigation

Vision-based mobile robot navigation is a relatively recent addition to the VISIONS research group at UMass. We have acquired a mobile robot that will enable us to develop a testbed for many of the vision algorithms we have and continue to develop. The robot is to be operated both indoors and out, providing a wide variety of scenes for analysis.

The UMass Autonomous Robot Architecture (AuRA) is being developed to support this research effort. It incorporates both global and reflexive schema-based path planning strategies and utilizes a priori knowledge stored in long-term memory, when available, to assist the vehicle's attainment of its navigational goals.

The chief navigational issues addressed include path following, landmark recognition for vehicle localization and obstacle avoidance. A new fast line finding algorithm is being used for hall and sidewalk navigation and will also be used for localization purposes. A depth-from-motion algorithm developed by Bharwani, Hanson and Riseman [8] is nearly completed and will be used initially for obstacle avoidance. It can also provide information for landmark identification when coupled with top-down knowledge of expected landmark locations. A new fast region segmentation algorithm has found potential application in both path following and vehicle localization. A description of all these algorithms and their use within AuRA can be found in [4] included in these proceedings.

Path planning is handled at two levels. First, the computation of a global path is conducted based on information stored in long-term memory in the form of a meadow-map. An A* search algorithm capable of dealing with the multiple terrain types found in the map is used to determine the initial route. Then information contained within the map is used to provide appropriate motor behaviors (motor schemas) to enable the robot to attain its navigational goals. Multiple concurrent processes, developed only in simulation thus far, provide the velocity vectors that constrain the robot's motion. Motor schemas afford a relatively straightforward mechanism, using a potential field methodology, for the combination of the outputs of individual motor tasks. These can readily reflect the uncertainty of the perceived environmental objects.

Our Gould system's pipeline parallel processing capabilities is currently being used for rapid application of look-up tables in both the line finding and region extraction algorithms. The acquisition of parallel hardware, a sequent multiprocessor, will decrease the processing time required for both vision and motor tasks is expected to enhance the real-time capabilities of the mobile robot project. When the UMass Image Understanding Architecture [43] is complete, much of the VISIONS system can be migrated directly into AuRA for real-time visual perception.

The successes in actual robot experimentation to date are modest. Successful navigation of both an outdoor sidewalk and an indoor hall using the line-finding algorithm has been achieved. The algorithm is quite robust working with (unchanging) environments in the presence of significant path edge discontinuities (doorways, vehicle tracks, clutter etc.). Obstacle avoidance on vehicle runs has been handled using ultrasonic data thus far. Dead-reckoning information is used minimally in our system as our goal is to serve as a testbed for vision algorithms.

Short-term goals include the finalization of the depth-from-motion algorithm in a form that is useful for obstacle avoidance applications. This algorithm is in the process of being transferred to the Carnegie-Mellon University vehicle navigation prospect (see Motion analysis section of this paper). Our vehicle should be able to navigate cluttered hallways and sidewalks solely using visual data. Installation of a recently acquired UHF transmitter link should be completed soon, allowing the vehicle a greater range than it currently has in its tethered form.

A hierarchical planning system consisting of a mission planner, navigator and pilot is being constructed to handle the task of path planning in both indoor and outdoor environments. Terrain features are taken into account in the determination of the best path for the mobile vehicle. The representations used will include a partial internal model of the environment. This enables the navigator to take advantage of a priori knowledge of the world while the pilot handles unanticipated and unmodeled obstacles as required.

Different path optimization strategies can be used based upon the mission's needs. Whether the safest path, shortest path, or some other metric constitutes the best path will depend on several factors. These would include the nature of the mission, the terrain to be traversed, temporal constraints, energy levels, positional uncertainty, etc. By modeling the free space of the vehicle's world expressly and tying relevant symbolic information to these "meadows", multiple factors are available for path-planning heuristics.

Possibly conflicting sensory input will have to be reconciled using "short-term memory" representations. The meadow map used for navigation will provide regions for instantiation based upon the robot's current position. Information from vision, ultrasonic sensors and positional sensors will be stored in this representation with associated certainty factors that will be altered based upon concurring or contradictory sensor input. This architecture will be sufficiently open-ended to allow the integration of additional sensor modalities (e.g. laser rangefinder, inertial guidance) as they become available.

Spatial and rotational uncertainty regarding the vehicle's position and orientation will be expressly modeled. The resulting spatial error map will be used to guide visual interpretation, windowing the image to reduce the time required for sensory processing. The sensory interpretations then will be used to reshape and reduce the spatial uncertainty map. The feedback provided by the sensors thus restricts the possible positions and orientations of the vehicle, while the probable location of the vehicle is used to guide sensory processing.

Homeostatic control (maintenance of the robot's own internal environment) is another research area. When mobile vehicles become capable of entering hazardous environments and covering longer distances without human monitoring, the status of the robot's energy levels, temperature, and other relevant variables can and should significantly affect planning and action. Through the use of internal sensors (in contrast to environmental sensors), surveillance of the internal state of the robot can be maintained. The information can then be used as necessary to change parameters for motor power consumption, heat production, etc., as well as provide data to the planner for decision making. Any vehicle purported to be "autonomous" must address this issue.

Many of the issues involved in the mobile vehicle research can be seen as complementary to those of other areas in our vision and robotics groups. The use of perceptual and motor schemas in the proposed vehicle architecture exploits many of the concepts used in both the VISIONS scene interpretation group and the work being done for the Laboratory for Perceptual Robotic's distributed programming environment. Multisensor integration, certainly crucial for the vehicle's domain, will only benefit from the work being done on the integration of vision, touch and force sensing.

5 Image Understanding Architecture (IUA)

UMass is designing and constructing a highly parallel architecture for computer vision with the goal of achieving real-time processing rates for a knowledge-based system approach to low, intermediate and high level image interpretation tasks. The project involves a joint design and implementation effort with the Hughes Research Laboratory. Our Image Understanding Architecture consists of three tightly coupled layers that correspond to three levels of abstraction. These layers are the Content Addressable Array Parallel Processor (CAAPP) for low-level processing which is a mesh connected array, the Intermediate and Communication as Associate Processor (ICAP) for intermediate vision, and the Symbolic Processing Array (SPA) for high-level processing. Attached to the SPA is a host processor.

As we have previously argued [34,43], an effective computational environment for image understanding requires tight coupling between the portions of the processor responsible for low-, intermediate-, and high-level processing [23,43,34]. In the IUA, the requirements of high-speed, fine-grained bi-directional communication and control is achieved using associative processing techniques to implement three very general processing/communcation capabilities:

1. Global Broadcast/Local Compare
2. Some/None Response
3. Count Responders

The CAAPP is 512 x 512 square grid array of 1-bit serial processors intended to perform low-level image processing tasks. The intermediate level is implemented by the Intermediate and Communications Associative Processor (ICAP). The ICAP is also a square grid associative array, of more powerful processing elements; the ICAP is a 64 by 64 array of 16-bit processors. Each ICAP cell is associated with an 8 by 8 tile of CAAPP cells, to which it has access. The SPA processors are powerful, general purpose microprocessors intended for performing high-level symbolic operations, and for controlling sub-array processing in the ICAP and CAAPP arrays. To the SPA, the lower levels appear as an intelligent database that is part of a shared global memory.

5.1 Associative Processing

Associative processing is a technique whereby the processors of the array have the ability to compare sets of data broadcast from a central controller to their own local data. They can then conditionally process both local data and broadcast data based on the results of those tests. Associative processing can best be understood by example, here a single controller (a teacher) interacting with an associative array (a class of students) [20]. If the teacher needs to know if any student in a class has a copy of a particular book the teacher can simply state, "If you have the book, raise your hand." The students each make a check, in parallel, and respond appropriately. This corresponds to a broadcast operation of a controller and a local comparison operation at each pixel in an array, to check for a particular value. Both operations assume that the local processors have some "intelligence" to perform the comparison.

Query and response is just the first part of associative processing, representing a content addressable ("If your hand is up, I'm talking to you.") scheme. To perform associative processing, we must be able to conditionally generate tags based on the value of data and use those tags for further processing. As processing continues only sub-sets of the pixels are involved in any particular operation, but those pixels are operated on in parallel.

The ability to associate tags with values is half the battle for high speed control. We also need to get responses back from the array quickly. Forcing the teacher to sequentially ask each student if they have their hand up defeats the process. The teacher can see immediately if any of the students have their hands up, and can quickly count how many do. Similarly, a Some-response/No-response (Some/None) wire running though the pixel array allows the controller to immediately determine properties about the data in the array, and therefore the state of processing in the array *without looking sequentially at the data values themselves.*

Additionally, fast hardware to perform a count of the responders allows the controller to see summary information about the state of the data in the array. We can write programs that can *conditionally* perform operations based on the state of the computation. By using the properties of the radix representation of numeric values in the array we can use the counting hardware to sum the values in the array. The ability to sum values gives us the power to compute statistical measures such as mean and variance.

These examples of students and pixels illustrate the power of associative processing. We use associative processing as our paradigm of communication in the upwards direction and control in the downwards direction between each pair of levels in the hierarchy. We broadcast criteria for selecting pixels, or regions, or symbolic tokens for selective processing. In this way the higher levels of processing control the lower levels. We test and/or count the response that comes after processing data to allow conditional branching for the next step of processing in a given algorithm. Thus the lower levels provide feedback to higher levels.

5.2 Current Status

At present we are building a 1/64th scale demonstration prototype of the IUA. This is scheduled for completion in early 1988 and will include 4096 CAAPP cells, 64 ICAP cells, a single SPA processor and global controller. The entire prototype will plug into a single-user workstation that will serve as a host.

The prototype is being constructed in 2 micron CMOS technology and will physically consist of a 16-by-20 inch motherboard with 83 daughterboards, 80 of which the daughterboards are 4 by 2.5 inches and the remaining three are 20 by 2.5 inches in size. Of the 80 small daughterboards, 16 are for clock distribution and signal buffering; the other 64 contain the ICAP and CAAPP processors and their memories. The three larger daughterboards provide the controller interface, feedback concentration, and ICAP communications network switching. The motherboard also includes a dual-ported frame buffer memory that allows simultaneous image input and output at video frame rate.

Each processor daughterboard will contain a single custom VLSI chip, a TMS320C25, 256K bytes of static RAM, 384K bytes of dual-ported dynamic RAM, and tri-state bus buffers. The single custom chip holds the 64 CAAPP processors with their local memories, the backing store controller, a refresh controller for the dynamic RAM, and arbitration logic for the various devices that must access the bus of the associated ICAP processor. The custom VLSI chip is currently undergoing fabrication through the MOSIS facility. A first run of the complete custom chip is scheduled for Summer of 1987. Total power dissipation for a processor daughterboard is estimated at approximately 5 watts.

Our software simulator is being re-written to run on an Odyssey signal processing co-processor board in a Texas Instruments Explorer. The Odyssey allows a direct emulation of the ICAP processor and greatly improves the execution times of CAAPP simulations over our VAX-based simulator. The Odyssey simulator will also permit us to closely mimic the interactions of the three processing levels down to the signal level. The Odyssey simulator will initially provide the capability of a single IUA daughterboard, and will eventually be extended to simulate one motherboard.

A VAX-based high-level emulator is also planned for development. Whereas the Odyssey simulator is designed to allow an assembly language level of programming, the VAX emulator will be the vehicle of choice for researchers who wish to get an idea of how the user-level IUA environment will behave. The emulator will sacrifice low-level accuracy in favor of greater speed. For example, the emulator will be restricted to 8, 16 and 32 bit arithmetic, thereby avoiding the bit-serial methods that are actually used in the CAAPP but are very slow in simulation.

Beyond simply testing our hardware design, our ultimate goal for the prototype is to provide a powerful interim development environment for image understanding parallel processing research. A simulated parallel processor is simply too slow to permit any significant amount of experimentation. Once our prototype is up and running, we will be able to accomplish more in the first ten seconds of execution time than we have been able to do in our previous five years of simulation.

Because having this much processing power in a box the size of a personal computer is so attractive, we have designed our prototype to be easily reproducible for a reasonable cost. It has also been designed to be easily adapted to different host systems. We thus hope that it will be possible to construct several copies of the small scale system so that it can be available to a number of researchers prior to construction of the full scale machine.

6 Motion Analysis

Our research in motion analysis has continued with a blend of theoretical and experimental investigations. There has been a concentration on the development of techniques that will find practical use in mobile vehicle navigation. In particular, we are in the process of transferring a motion algorithm from UMass to CMU for recovery of depth under known motion; we expect it to be useful for both obstacle avoidance and landmark recognition. Let us now discuss some of these efforts in somewhat more detail.

Our past motion research concentrated in the recovery of sensor motion parameters from analysis of two images obtained via a sensor in motion. This work was reported in the Ph.D. dissertation research of Lawton [31,32] and Adiv [1]. More recently Pavlin [38] has evaluated the Lawton algorithm for translational motion and determined that the algorithm can be applied effectively with analysis of only 8 to 16 image points between frames if the sensor is pointed approximately in the direction of sensor motion. In addition he has speeded up the algorithm and made it more robust by improving the FOE search algorithm. This was accomplished by computing the error measure for the assumed FOE from a sparser sampling of the visual field (kor a more restricted area if constraints on the possible location of the FOE is available). Then, a smooth surface is fit to the error values at those points and the computed minimum of this surface is used to focus the search in the next step of an iterative search process.

Bharwani et al [8,9,10] has continued to develop an algorithm that will compute increasingly more accurate depth information from a sequence of frames derived via approximately known translational motion of the sensor. This algorithm is intended to be applied after FOE recovery using the Lawton-Pavlin algorithm, or when vehicle instrumentation supplies sensor motion. The algorithm matches points between frames up to some match resolution, computes a depth range for the environmental point, and then uses this information to predict a smaller search window in future frames, which then can be searched with finer match resolution and consequently more accurate depth. An important characteristic of this algorithm is that the temporal depth refinement can be applied at a constant computational rate and therefore is well-suited for robot navigation. Since the last report on this algorithm, [10] it has been modified to include the implications of Snyder's theoretical treatment of uncertainty [41] discussed below. Because the FOE and an image point/feature in the first frame actually have an uncertainty region that is two dimensional (at a minimum due to digitization error), the search region must also be two-dimensional. This modification has improved the robustness of the algorithm. In addition the shape of the error surface [3] can be used to dynamically control the resolution of the depth refinement process to experimentally measurable limits.

The two algorithms, FOE recovery and temporal depth refinement, are being packaged into a motion analysis subsystem for use in both the UMass and CMU mobile vehicle efforts. The goal is the analysis of an ongoing sequence of frames from a vehicle in motion to determine obstacles in the path of motion. At CMU it is hoped that this subsystem will operate effectively at a range beyond the useful range of the ERIM sensor (40 foot limit). There are three very general stages of processing that will be briefly discussed. First, frames must be registered since the camera will not be independently stabilized and therefore jerks, bumps, rocking, etc. will introduce local random translational and rotational motion between frames even when the vehicle is undergoing approximate pure global translation. Registration is currently our major problem, and thus for only having a simple registration scheme involving the slection of distinctive points (high contrast and high curvature) that are at a great distance (near horizon) and thus will allow subtraction of the rotational component. Then the FOE will be recovered via the Lawton-Pavlin algorithm using a small number of distinctive points, say 8, in the foreground (10-40 feet). Then the depth of distinctive points in the path of the vehicle will be computed. Finally, either point sets that imply vertical surfaces, or individual points that are not consistent with lying on a planar road surface will be flagged for higher level navigational attention.

Snyder [41,42], has theoretically examined the problem of uncertainty of image measurements in correspondence-based techniques, and their impact in stereo and motion analysis. The location of image features or points are often determined only approximately due to the effect of processing with a window (e.g. as in computing interest operators or using convolution windows) or the result of more complicated processes as in FOE recovery. At a minimum there is sub-pixel uncertainty ($\pm 1/2$ pixel) due to digitization. Uncertainty in such image locations leads to uncertainty in the recovery of depth from both stereo and motion, defines limits to the effectiveness of recovering depth of environmental points or detecting the presence of independently moving objects, and provides the means to determine the relative efficacy between stereo and motion analysis in varying situations. The analysis provides strategies for intelligently controlling the application of stereo and motion algorithms and determining uncertainty ranges for the results that are extracted.

Glazer's recently completed thesis [21] presents an approach to motion detection using multi-resolution methods in a hierarchical processing architecture. Two motion detection algorithms are developed and analyzed. The hierarchical correlation algorithm utilizes a coarse-to-fine control strategy across the resolution levels and overcomes two disadvantages of single-level correlation: large search areas requiring expensive searches and repetetive image structures which cause incorrect matches. The hierarchical gradient-based algorithm [22], generated over low-pass image pyramids, extends single-level gradient algorithms to the computation of large displacements. Within each level the next refinement of the displacement field is obtained by combining a local intensity constraint and a global smoothness constraints. The mathematical formulation involves the minimization of an error functional consisting of two terms, corresponding to the intensity and the smoothness constraints mentioned above. The minimization problem is solved using the finite-difference approach which leads to a multi-resolution relaxation algorithm. A formal analysis of the hierarchical gradient algorithm is presented, including the basic equations for computing a refined disparity vector, the discrete representations and computations for solving these equations, and a geometric interpretation of the resulting relaxation algorithm. The experimental results show that the two algorithms have comparable accuracy and a cost analysis shows that the hierarchical gradient algorithm is less costly.

In his recently completed doctoral dissertation [2] Anandan provides a unified framework for extracting a dense displacement field from a pair of images, as well as an integrated system which is based on a matching approach. This framework appears to be sufficiently general to encompass both gradient-based and correlation matching approaches. It consists of a hierarchical scale-based matching scheme using bandpass filters, orientation-dependent confidence measures, and a smoothness constraint for propagating reliable displacements. His integrated system for the extraction of displacement fields uses the minimization of the sum-of-squarred-differences (SSD) as the local match-criterion, computes confidence measures based on the shape of the SSD surface, and formulates the smoothness assumption as the minimization of an error functional, and overcomes many of the difficult problems that exist in other techniques. The error functional consists of two terms: one of which is called the approximation error, measuring how well a given displacement field approximates the local match estimates, while the other is called the smoothness error, measuring the global spatial-variation of a given displacement field. The finite-element method is used to solve the minimization problem. The approach also gives information for extracting occlusion boundaries in some situations.

Anandan has also shown that the functional minimization problem formulated in his matching technique converges to the minimization problem used in gradient-based techniques (e.g. Glazer's technique mentioned above). In particular, by relating an approximation error functional used in his matching approach to the intensity constraints used in the gradient-based approaches, he explicitly identifies confidence measures which have thus far been implicitly used in the gradient-based approach. Finally, he suggests the ways that algorithms operating on a pair of frames can be developed into multiple-frame algorithms, while discussing their relationship to spatio-temporal energy models.

7 Low-Level Vision

While low-level vision is not the main focus of our research program, almost any large group working on intermediate and high-level computer vision will be engaged in some aspects of low-level vision to support the other efforts. There are several basic segmentation algorithms that our knowledge-based vision research relies upon: histogram-based region segmentation [30,35], straight line extraction [14], and more recently an algorithm for grouping nearby co-linear edges [11]. Analysis of the output of these algorithms has led to several additional investigations. Interesting work is being directed towards edge and line algorithms, as well as texture extraction.

The output of both of our straight line algorithms has made it very obvious that short lines are a very effective mechanism for extracting textured areas and texture descriptions. Since each line has a set of attributes including orientation, length, contrast, etc. they can be filtered or grouped in terms of a variety of features. This may lead to interesting ways to directly extract and characterize textured areas. Alternatively, lines may be used to provide regions with texture characteristics.

An algorithm for grouping edges into curved line segments has continued to be developed and has yielded some promising results [18]. It may be integrated with the straight line algorithm by choosing the output from each representation that is most appropriate. Thus, parameterized curves might replace a piecewise linear sets of edges in the intermediate representation that is examined by knowledge-based interpretation processes.

The algorithm for extracting straight lines by grouping on gradient orientation [14] is being expanded by Reynolds to work on color information rather than intensity. Thus, areas of an image with similar intensity but different color might not be detected by the original algorithm. However, by computing orientation in 3-dimensional color space, edges can be labelled with both their orientation and the colors on either side of the edge. In fact this leads to a straight line extraction algorithm where the line segments represent edges which delimit the boundary between regions of approximately constant color; i.e. instead of a line segment being defined by a gradient magnitude threshold or uniform gradient orientation, the constancy of color contrast across the boundary can also be employed. An additional effort to group similar color edges into textured areas is also being investigated.

Finally, Kitchen and Malin [27] have completed a study of the effect of spatial discretization on the magnitude and direction response of various simple edge operators. They investigate the errors as the true subpixel location of an ideal step edge is varied. Their results show a potentially significant variation can occur in edge magnitude and orientation. They include suggestions for possible improvements of edge operators based upon their techniques.

8 References

REFERENCES

[1] G. Adiv, "Interpreting Optical Flow", Ph.D. Dissertation, Computer and Information Science Department, University of Massachusetts at Amherst, September 1985. Also COINS Technical Report 85-35.

[2] P. Anandan, "Measuring Visual Motion From Image Sequences", Ph.D. Dissertation, University of Massachusetts at Amherst, 1987.

[3] P. Anandan, "A Unified Perspective on Computational Methods for the Measurement of Visual Motion", *Proceedings of the DARPA Image Understanding Workshop*, Los Angeles, CA, January 1987.

[4] R. C. Arkin, E. M. Riseman, and A. R. Hanson, "AuRA: An Architecture for Vision-Based Robot Navigation", *Proceedings of the DARPA Image Understanding Workshop*, Los Angeles, CA, January 1987.

[5] R. Belknap, E. Riseman, and A. Hanson, "The Information Fusion Problem and Rule-Based Hypotheses Applied To Complex Aggregations of Image Events", COINS Technical Report, University of Massachusetts at Amherst, in preparation, 1987.

[6] R. Belknap, E. Riseman, and A. Hanson, "The Information Fusion Problem and Rule-Based Hypotheses Applied to Complex Aggregations of Image Events", *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Miami, FL, June 22-26, 1986, pp. 227-234.

[7] P.J. Besl and R.C. Jain, "Segmentation Through Symbolic Surface Description", *Proceedings of CVPR86*, Miami FL, June 1986, pp. 77-85.

[8] S. Bharwani, E. Riseman, and A. Hanson, "Multiframe Computation of Accurate Depth Maps Using Uncertainty Analysis", forthcoming technical report, Computer and Information Science Department, University of Massachusetts at Amherst.

[9] S. Bharwani, E. Riseman, and A. Hanson, "Refinement of Environmental Depth Maps over Multiple Frames", *Proceedings of the IEEE Workshop on Motion: Representation and Analysis*, Charleston, SC, May 7-9, 1986, pp. 73-80.

[10] S. Bharwani, E. Riseman, and A. Hanson, "Refinement of Environmental Depth Maps over Multiple Frames", *Proc. of the DARPA Image Understanding Workshop*, Miami Beach, FL, December 1985.

[11] R. Weiss and M. Boldt, "Geometric Grouping Applied to Straight Lines", *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Miami, FL, June 22-26, 1986, pp. 489-495.

[12] M. Boldt and R. Weiss, "Geometric Grouping Applied to Straight Lines", forthcoming technical report, Computer and Information Science Department, University of Massachusetts at Amherst.

[13] M. Brady, J. Ponce, A. Yuille, and H. Asada, "Describing Surfaces", *Proceedings of the 2nd International Symposium on Robotics Research*, Hanafusa and Inoue (Eds.), MIT Press, Cambridge, MA

[14] J. B. Burns, A. R. Hanson, and E. M. Riseman, "Extracting Straight Lines", *IEEE Transactions on Pattern Analysis and Machine Intelligence 8*, No. 4, July 1986, pp. 425-455. Also COINS Technical Report 84-29, University of Massachusetts at Amherst, December 1984.

[15] J. Callahan and R. Weiss, "A Model for Describing Surface Shape", *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, San Francisco, June 1985, pp. 240-245.

[16] J. Callahan, "Local Views of a Piecewise Smooth Surface", in preparation.

[17] A. P. Dempster, "A Generalization of Bayesian Inference", *Journal of the Royal Statistical Society*, Series B, Vol. 30, 1968, pp. 205-247.

[18] J. Dolan, G. Reynolds, and L. Kitchen, "Piecewise Circular Description of Image Curves Using Constancy of Grey-level Curvature", COINS Technical Report 86-33, University of Massachusetts at Amherst, July 1986.

[19] B. Draper, R. Collins, J. Brolio, J. Griffith, A. Hanson, and E. Riseman, "Tools and Experiments in the Knowledge-Based Interpretation of Road Scenes", *Proceedings of the DARPA Image Understanding Workshop*, Los Angeles, CA, January 1987.

[20] C. C. Foster, "Content Addressable Parallel Processors", Van Nostrand Reinhold, New York, 1976.

[21] F. Glazer, "Hierarchical Motion Detection", Ph.D. Dissertation, Computer and Information Science Department, University of Massachusetts at Amherst, 1987.

[22] F. Glazer, "Hierarchical Gradient-Based Motion Detection", *Proceedings of the DARPA Image Understanding Workshop*, Los Angeles, CA, January 1987.

[23] A. Hanson and E. Riseman, "The VISIONS Image Understanding System - 1986", in *Advances in Computer Vision*, (Chris Brown, Ed.), Erlbaum Press, 1987.

[24] A. Hanson and E. Riseman, "A Methodology for the Development of General Knowledge-Based Vision Systems", to appear in *Vision, Brain, and Cooperative Computation*, (M. Arbib and A. Hanson, Eds.), 1987, MIT Press Cambridge, MA. Also COINS Technical Report 86-27, University of Massachusetts at Amherst, July 1986.

[25] K. Kanatani, "The Constraints on Images of Rectangular Polyhedra", *PAMI*, 8, No. 4, pp. 456-463, 1986.

[26] Y. L. Kergosien, "La famille des projections orthogonales d'une surface et ses singularities", *C.R. Acad. Sc. Paris*, pp. 929-932, 1981.

[27] L. Kitchen and J. Malin, "The effect of spatial discretization on the magnitude and direction response of simple differential edge operations on a step edge. Part I: square pixel receptive fields", forthcoming technical report, Computer and Information Science Department, University of Massachusetts at Amherst.

[28] J. J. Koenderink, "What Does the Occluding Contour Tell us About Solid Shape?", *Perception*, vol 13, 1984, pp. 321-330.

[29] C. Kohl, A. Hanson, and E. Riseman, "A Goal-Directed Intermediate Level Executive for Image Interpretation", *Proceedings of the DARPA Image Understanding Workshop*, Los Angeles, CA, January 1987.

[30] R. R. Kohler, "A Segmentation System Based on Thresholding", *Computer Graphics and Image Processing*, **15**, 1981, pp. 319-338.

[31] D. T. Lawton, "Processing Dynamic Image Sequences from a Moving Sensor", Ph.D. Dissertation, Computer and Information Science Department, University of Massachusetts at Amherst, 1984. Also COINS Technical Report 84-05.

[32] D. T. Lawton, "Processing Translational Motion Sequences", *Computer Graphics and Image Processing*, Vol 22, pp. 116-144, 1983.

[33] N. Lehrer, G. Reynolds, and J. Griffith, "Initial Hypothesis Formation in Image Understanding Using an Automatically Generated Knowledge Base", *Proceedings of the DARPA Image Understanding Workshop*, Los Angeles, CA, January 1987.

[34] S. Levitan, C. Weems, A. Hanson, and E. Riseman, "The UMass Image Understanding Architecture", in *Pyramid Multi-Computers*, (Leonard Uhr, Ed.), Academic Press, New York, 1987.

[35] P. A. Nagin, "Studies in Image Segmentation Algorithms Based on Histogram Clustering and Relaxation", COINS Technical Report 79-15, University of Massachusetts at Amherst, September 1979.

[36] H. Nakatani, R. Weiss, and E. Riseman, "Application of Vanishing Points to 3D Measurement", *SPIE International Symposium on Optics and Electro-Optics*, 1984.

[37] H. Nakatani, "Reconstruction of Three-Dimensional Shape Using Pictorial Depth Cues", Ph.D. Thesis, Osaka University, 1986.

[38] I. Pavlin, E. Riseman, and A. Hanson, "A Translational Motion Algorithm Using Hierarchical Search with Smoothing", forthcoming technical report, Computer and Information Science Department, University of Massachusetts at Amherst.

[39] G. Reynolds and R. Beveridge, "Searching for Geometric Structure in Images of Natural Scenes", *Proceedings of the DARPA Image Understanding Workshop*, Los Angeles, CA, January 1987.

[40] G. Shafer, "A Mathematical Theory of Evidence", Princeton University Press, 1976.

[41] M. Snyder, "Uncertainty Analysis in Image Measurements", forthcoming technical report, Computer and Information Science Department, University of Massachusetts at Amherst.

[42] M. Snyder, "Uncertainty Analysis in Image Measurements", *Proceedings of the DARPA Image Understanding Workshop*, Los Angeles, CA, January 1987.

[43] C. Weems, S. Levitan, E. Riseman, and A. Hanson, "The Image Understanding Architecture", *Proceedings of the DARPA Image Understanding Workshop*, Los Angeles, CA, January 1987.

[44] R. Weiss and P. Giblin, "On the Reconstruction of Surfaces From Their Profiles", *Proceedings of the DARPA Image Understanding Workshop*, Los Angeles, CA, January 1987.

[45] T.E. Weymouth, "Using Object Descriptions in a Schema Network For Machine Vision", Ph.D. Dissertation, Computer and Information Science Department, University of Massachusetts at Amherst. Also COINS Technical Report 86-24, University of Massachusetts at Amherst, 1986.

Recent Progress of the Rochester Image Understanding Project

Jerome A. Feldman and Christopher M. Brown

Computer Science Department
University of Rochester
Rochester, New York 14627

1. Robust Vision Operators

1.1. Parameter Networks and the Hough Transform

One of the most difficult problems in vision is segmentation. Recent work has shown how to calculate intrinsic images (e.g., optical flow, surface orientation, occluding contour, and disparity). These images are distinctly easier to segment than the original intensity images. Such techniques can be greatly improved by incorporating Hough methods. The Hough transform idea has been developed into a general control technique. Intrinsic image points are mapped (many to one) into 'parameter networks' [Ballard 1984]. This theory explains segmentation in terms of highly parallel cooperative computation among intrinsic images and a set of parameter spaces at different levels of abstraction.

Earlier work on the Hough transform has led in three directions.

1) Research toward a theory of cache accumulator arrays [Loui *et al.* 1983; Brown & Feldman 1983]

2) Experiments with complementary HT and cache management strategies [Brown *et al.* 1983].

3) Hardware (VLSI) designs for HT vote caches [Sher & Tevanian 1983].

Recent efforts extend these ideas and are moving them into the parallel computing environment of the BBN Butterfly.

1.2 Bayesian Detectors

A recent extension of our work on low level operators involves the exploration of optimal operators for early vision. This is one aspect of our increased effort on formal evidence theory and its application in intelligent systems [Kyburg 1985; Sher 1985].

1.3 High Level Planning

In general, problem solvers cannot hope to create plans that are able to specify fully all the details of operation beforehand and must depend on run-time modification of the plan to insure correct functioning. The run-time planning idea becomes particularly important when different plan segments are being explored concurrently. These communicating segments may require sophisticated actions e.g. (do PLAN$_x$ until PLAN$_y$). These issues were studied by [Russell 1984] in the context of a cooperative planning and execution system for manipulation tasks. A recent effort [Ballard 1984] is examining robot planning from a task frame perspective.

2. Computing with Connections

We are continuing our interest in problem-scale parallelism, both as a model of animal brains and as a paradigm for VLSI and parallel computing [Feldman *et al.* 1985]. Work at Rochester has concentrated on connectionist models and their application to vision. The framework is built around computational modules, the simplest of which are termed p-units. We have developed their properties and shown how they can be applied to a variety of problems. We have also established powerful techniques for adaptation and change in these networks.

Recent theoretical work has examined the convergence properties of these networks [Feldman 1985d] and their representational capabilties [Feldman 1986c].

The second generation connectionist system has now been in active use for well over two years. Compatible versions run on the VAX systems, SUN workstations and the BBN Butterfly multi-computer. As expected, the Butterfly version makes excellent use of the available parallelism [Fanty 1986]. The parallel simulator including a new general graphics interface, will be released to the public in early 1987.

The enhanced computational facilities have facilitated applications work with connectionist models. Among the problems addressed are parsing [Fanty 1986], lexical disambiguation [Cottrell 1985] and semantic inference [Shastri 1985]. A major current effort explores learning and adaptation in structured connectionist networks.

3. Parallel Computation in Image Understanding

It has been clear for many years that practical solutions to image understanding problems will require parallel computation. Great progress has been made in early vision and in the development of machines specialized for these computations. Intermediate and recognition level vision are more complex; it is much less obvious how to compute them in parallel. This has been a major focus of the Rochester effort for several years.

Our approach to parallelism in Image Understanding is based on the belief that general purpose (MIMD) machines will work out best for the full range of vision problems [Feldman 1985b]. Our work has taken a major step forward with the successful installation of a 128-processor BBN Butterfly computer. In addition to the massively parallel approaches mentioned above, we are looking at conventional vision algorithms and at general purpose parallel programing methodologies [Brown *et al.* 1985].

The first protoype of a fully parallel integrated vision system has been completed [Cooper & Hollbach, this volume]. This program for recognizing structures built of tinker toys is complete from input to recognition and every step is parallelizable. Various experiments are underway to test alternative realizations of the components of the system and to extend its capabilities.

The references show two new technical report series. The *Butterfly Project Report* and *Hierarchical Process Composition Project Report* series document our work on the Butterfly Parallel Processor and in the HPC computational model. Papers in the references by Aloimonos, Ballard, Bandopadhay, Brown, Olson, Bukys, Fowler, Chou, Cooper, Hinkelman, Hollbach, Narayanan, Sher, and Swain give details of work mentioned below.

4. Motion and Real-time AI

During 1986, Aloimonos' work centered on the robust and reliable computation of intrinsic images, or physical parameters of the scene. He invented several new techniques, and his method has been to add information sources rather than to rely exclusively on a priori constraints (such as smoothness). His work has mainly been in the domains of multiple frame vision (stereo, motion) and in texture. Bandopadhay was also working in the domain of motion. His work has been to apply clustering to the motion segmentation and egomotion problem, and to notice that proprioceptive feedback from tracking stationary points can work with vision to make the egomotion calculations easier.

The tracking work is the scientific motivation for certain robotic hardware designed and built at Rochester, which consists of two cameras on a "robot head". With this setup we are investigating real-time vision. A pilot project by Dave Coombs and Brian Marsh has produced a software framework and a working program that tracks multiple moving colored objects in a scene. This work emphasizes multi-resolution processing, focus of attention, and real-time and priority job scheduling, but does not mechanically move the cameras to implement its active attention control. Tom Olson is continuing this work. With the acquisition of new and faster host computers, new and faster low-level image analysis hardware and hardware upgrades for our Butterfly Parallel Processor, we hope to build a multi-camera system that can track and analyse multiple objects in real time in navigational and robotic contexts.

4.1 BIFF: A Butterfly Vision Library

Tom Olson and Liud Bukys have constructed a parallel version of the IFF vision library written at the University of British Columbia under the direction of Prof. Havens. IFF is a file organization for images, and an associated set of image processing and vision utilities, something like SPIDER or GIPSY. IFF programs are written as UNIX filters, and the system uses UNIX pipes to concatenate operations. This is a slow way to go about things but is very modular and good for interactive use. BIFF, the parallel version, is much faster, both through capitalizing on the innately parallel nature of many low-level vision operations, and through use of the large memory on the individual butterfly nodes to achieve "in-core" files that can be passed from process to process quickly through memory mapping. We expect BIFF to be a useful tool and to expand in the future.

4.2 Segmentation with the Uniform System

Tom Olson has constructed an advanced program under the Uniform System to do segmentation. We are interested in the general problem of combining the outputs of low-level vision processes to produce robust interpretation of large classes of input images. In addition, we want solutions that make efficient use of large-scale parallel hardware. In order to study these issues we have chosen a particular well-studied problem (2-d segmentation) for implementation othe BBN Butterfly Multiprocessor. To date, we have been more concerned with communications and systems aspects of the combination than with the mathematical aspects of cooperating constraints or evidence combination.

The program works by recursively splitting regions until all regions satisfy some termination criteria. Users of the system must provide a set of functions called *experts* which take as their argument some region and generate a proposed segmentation of that region. The user also provides a *reconciling* function which integrates a set of proposed segmentations into a single proposal, which the main program then executes. With minor changes, a large class of currently used segmentation algorithms can be fit into this model. Among its defects are that a) there is no provision for merging, and b) reasoning based on more than one region (e.g. based on connectivity) is forbidden. The current implementation has only one expert function, a grey-level histogram splitter loosely based on PHOENIX, the multispectral segmentor of Shafer and Kanade[1]. The program is well parallelized, using the Uniform System library to implement parallel loops. Users provide an initialized vector of pointers to the expert functions and the reconciler. The program makes use of BIFF (see above).

This segmentation program illuminated many issues of parallelization, load balancing, and models of computation for tightly coupled MIMD machines such as the Butterfly.

5. Multi-modal Segmentation

Paul Chou [this volume] is building a segmentation program that uses multiple modes of information (intrinsic images such as depth and local surface orientation, structured light, image intensity). His approach uses Markov Random Fields as a way of expressing the probability of local configurations of evidence. A method of combining likelihoods is used to do incremental evidence combination. So far the likelihoods have been provided by Dave Sher's operators (see below). Local evidence combination yields intermediate results that are combined by more global methods (for instance grouping processes and methods to fit parameterized surface models to data) into segments (descriptions of three-dimensional surface patches).

6. Bayesian Vision Operators

Dave Sher [this volume] has developed a method to generate operators tuned to image noise conditions, edge types, and a priori knowledge about boundary type frequencies. His work is foundational in that it shows how to construct tuned operators given whatever knowledge is available about the scene domain. Further, if there are several possible models of the world operating and it is unknown which is in fact true, Sher's mechanism allows for the combination of evidence from several operators, each tuned to one of the possible states of nature. Sher's operators are generalizations of usual edge operators in that they produce confidences in their own applicability, and general probabilities for features rather than decisions about features. His operators have been (and are being) tested on a variety of synthetic and natural images.

7. Parallel algorithms for Image Understanding

Rochester took part in the DARPA IU Benchmark exercise. A summary of the results will be provided at this IU workshop by Prof. Azriel Rosenfeld, and four Butterfly Project Reports document the methods and results in more detail. Using several different programming environments developed under DARPA funding under the Autonomous Vehicle program, all the benchmarks were implemented in a few weeks. The work to develop programming environments for parallel vision algorithms is continuing successfully.

References

Refereed Papers

Allen, J.F. and P.J. Hayes, "A common-sense theory of time," *Proc., 9th Int'l. Joint Conf. on Artificial Intelligence*, Los Angeles, CA, August 1985; TR 180 (longer version), Computer Science Dept., Univ. Rochester, to appear, 1986.

Allen, J.F. and D.J. Litman, "Plans, goals and natural language," *1986-87 Computer Science and Engineering Research Review*, Computer Science Dept., Univ. Rochester, October 1986; *IEEE Special Issue on Natural Language Processing* 74, No. 7, July, 1986, 939-947.

Allen, J.F. and R. Pelavin, "Using counterfactuals to represent actions and their effects," to appear, *IEEE Special Issue on Knowledge Representation*, 1986.

Aloimonos, J. and C.M. Brown, "Perception of structure from motion, 1) Optical flow vs. discrete displacements, II) Lower bound results," *IEEE Conf. on Computer Vision and Pattern Recognition*, Miami Beach, FL, June 1986.

Ballard, D.H., "Parameter nets," *Artificial Intelligence*, 22, 235-267, 1984.

Ballard, D.H., "Task Frames in Robot Manipulation," *Proceedings*, AAAI-84, August, 1984.

Brown, C.M., "Color, texture, shape and motion," submitted, *Pattern Recognition Letters*, September 1985.

Brown, C.M. and J. Aloimonos, "Perception of structure from motion, 1) Optical flow vs. discrete displacements, II) Lower bound results," *IEEE Conf. on Computer Vision and Pattern Recognition*, Miami Beach, FL, June 1986.

Brown, C.M., J. Aloimonos, M. Swain, P. Chou, and A. Basu, "Texture, contour, shape and motion," submitted, *Pattern Recognition Letters*, September 1985.

Cottrell, G.W., "Parallelism in Inheritance Hierarchies with Exceptions," Proceedings, Int'l. Joint Conf. on Artificial Intelligence, Los Angeles, CA, 194-202, August 1985.

Cottrell, G.W., "A Conectionist Approach to Word Sense Disambiguation," Ph.D. thesis, Computer Science Dept., Univ. Rochester, April 1985; also TR145, Computer Science Dept., Univ. Rochester, May, 1985.

Ellis, C.S., "Concurrency and linear hashing system," *1985-86 Computer Science and Engineering Research Review*, Computer Science Dept., Univ. Rochester, September 1985.

Ellis, C.S., "Distributed data structures: A case study," *IEEE Trans. on Computers C-34*, 1178-1185, 1985.

Ellis, C.S. and R.A. Floyd, "Work in progress: Distributed and multiprocessor file systems," *SOSP 10*, December 1985.

Feldman, J.A., "Connectionist models and parallelism in high level vision, *CVGIP 31*, Special Issue on Human and Machine Vision, 178-200, 1985a.

Feldman, J.A., "Connections -- Massive parallelism in natural and artificial intelligence," *BYTE*, 277-284, April 1985b.

Feldman, J.A., "Four Frames Suffice: A Provisional Model of Vision and Space," Behavioral and Brain Sciences, 8, 265-289, June 1985c.

LeBlanc, T.J., "Shared memory versus message-passing in a tightly-coupled multiprocessor: a case study," *Proc., 1986 Int'l. Conf. on Parallel Processing*, October 1986. Recipient of Distinguished Presentation Award.

LeBlanc, T.J. and S.A. Friedberg, "HPC: a model of structure and change in distributed systems," *IEEE Trans. Computers C-34*, 12, 1114-1129, 1985.

LeBlanc, T.J., "Shared memory versus message-passing in a tightly-coupled multiprocessor: a case study," to appear, *IEEE Trans. Computers*; also, Butterfly Project Report 3, Computer Science Dept., Univ. Rochester, January 1986.

Scott, M.L., "Language support for loosely-coupled distributed programs," *IEEE Trans. Software Engineering*, SE-13, No. 1, 88-103, Jan. 1987.

Shastri, L., "Evidential Reasoning in Semantic Networks - A Formal Theory and its Parallel Implementation," Ph.D. thesis, Computer Science Dept., Univ. Rochester, July 1985; also, TR166, Computer Science Dept., Univ. Rochester. September 1985.

Sher, D.B., "Optimal likelihood generators for edge detection under Gaussian additive noise," *1986-87 Computer Science and Engineering Research Review*, Computer Science Dept., Univ. Rochester, October 1986.

Vilain, M. and H. Kautz, "Constraint propagation algorithms for temporal reasoning," *Proc., AAAI-86*, Univ. Pennsylvania, 1986.

Technical Reports:

Allen, J.A., "Natural language lecture notes," Computer Science Dept., Univ. Rochester, January 1986.

Aloimonos, J., A. Basu, and C.M. Brown, "Coutour, shape and motion," *Proc., DARPA Image Understanding Workshop*, Miami, FL, December 1985.

Brown, C.M., "Artificial intelligence research on the Butterfly multiprocessor," *Proc., National Academy of Sciences Workshop on AI and Distributed Problem Solving*, Washington, DC, May 1985.

Brown, C.M., M. Curtiss, and D. Sher, "Bias and Noise in Hough Transform: Experiments," *Proceedings*, IJCAI-83, Karlsruhe, West Germany, August 1983.

Brown, C.M. and J.A. Feldman, "Statistical Questions Arising in the Use of Hough Techniques in Image Understanding," *Proceedings*, ONR Workshop on Statistical Image Processing and Graphics, Luray VA., May 1983.

Brown, C., R. Fowler, T. LeBlanc, M. Scott, M. Srinivas, L. Bukys, J. Costanzo, L. Crowl, P. Dibble, N. Gafter, B. Marsh, T. Olson, L. Sanchis, "DARPA parallel architecture benchmark study," Butterfly Project Report 13, Computer Science Dept., Univ. Rochester, October 1986.

Bukys, L., "Connected component labeling and border following on the BBN Butterfly parallel processor," Butterfly Project Report 11, Computer Science Dept., Univ. Rochester, October 1986.

Bukys, L. and T.J. LeBlanc, "Getting started with the BBN Butterfly parallel processor," Butterfly Project Report 1, Second Edition, Computer Science Dept., Univ. Rochester, October 1986.

Costanzo, J., L. Crowl, L. Sanchis and M. Srinivas, "Subgraph isomorphism on the BBN Butterfly multiprocessor," Butterfly Project Report 14, Computer Science Dept., Univ. Rochester, October 1986.

Crowl, L.A., "Chrysalis + +," Butterfly Project Report 15, Computer Science Dept., Univ. Rochester, December 1986.

Fanty, M., "A connectionist simulator for the BBN Butterfly Multiprocessor," Butterfly Project Report 2, Computer Science Dept., Univ. Rochester, January 1986; also TR164, Computer Science Dept., January 1986.

Feldman, J.A., "Neural representation of conceptual knowledge," TR189, Computer Science Dept., Univ. Rochester, June 1986.

Feldman, J.A., "Energy and the behavior of connectionist models," TR 155, Computer Science Dept., Univ. Rochester, November 1985.

Feldman, J.A., D.H. Ballard, C.M. Brown, and G.S. Dell, "Rochester connectionist papers: 1979-1985," TR 172, Computer Science Dept., Univ. Rochester, December 1985.

Feldman, J.A. and C.M. Brown, "Recent progress of the Rochester image understanding project," *Proc., DARPA Image Understanding Workshop*, Miami, FL, December 1985.

Feldman, J.A., J.L. McClelland, G. Bower and D. McDermott, "Connectionist models and cognitive science: Goals, directions and implications," position paper from NSF Workshop on Connectionist Modelling, 1986.

Friedberg, S.A., "Control of dynamic process structure," Hierarchical Process Composition Project Report 6, Computer Science Dept., Univ. Rochester, October 1986.

Friedberg, S.A., "HPC coding style guidelines," Hierarchical Process Composition Project Report 1, Computer Science Dept., Univ. Rochester, May 1986

Friedberg, S.A., "User process - HPC interface," Hierarchical Process Composition Project Report 3, Computer Science Dept., Univ. Rochester, October 1986.

Friedberg, S.A. and D.H. Pitcher, "HPC IPC implementation," Hierarchical Process Composition Project Report 2, Computer Science Dept., Univ. Rochester, June 1986.

Friedberg, S.A. and I. Rigoutsos, "Comments on Stony Brook MP," Hierarchical Process Composition Project Report 4, Computer Science Dept., Univ. Rochester, May 1986.

Hinkelman, E., "NET: A utility for building regular process networks on the BBN Butterfly parallel processr," Butterfly Project Report 5, Computer Science Dept., Univ. Rochester, September 1986.

Hollbach, S.C., "Tinker toy recognition from 2D connectivity," TR 196, Computer Science Dept., Univ. Rochester, October, 1986.

Kyburg, H.E. Jr., "Bayesian and Non-Bayesian Evidential Updating, TR139 (revised), Computer Science Dept., Univ. Rochester, July 1985.

LeBlanc, T.J. and J.M. Mellor-Crummey, "Debugging parallel programs with instant replay," Butterfly Project Report 12, Computer Science Dept., Univ. Rochester, September 1986.

LeBlanc, T.J., N.M. Gafter, and T. Ohkami, "SMP: a message-based programming environment for the BBN Butterfly," Butterfly Project Report 8, Computer Science Dept., Univ. Rochester, July 1986.

Litman, D.J., "Plan recognition and discourse analysis: an integrated approach for understanding dialogues," Ph.D. thesis and TR 170, Computer Science Dept., Univ. Rochester, September 1985.

Loui, R.P., J.A. Feldman and H.E. Kyburg, Jr., "Interval-Based Decisions for Reasoning Systems," Proceedings, AAAI Workshop on Probability and Uncertainty in AI., 1985.

Low, J.R., "Experiments with remote procedure call on the Butterfly," Butterfly Project Report 16, Computer Science Dept., Univ. Rochester, December 1986.

Narayanan, N.H. and C.M. Brown, "Parallel stereo correspondence on the Butterfly Multiprocessor," forthcoming Butterfly Project Report, Computer Science Dept., Univ. Rochester, to appear, 1986.

Olson, T.J., "Finding lines with the Hough transform on the BBN Butterfly parallel processor," Butterfly Project Report 10, Computer Science Dept., Univ. Rochester, September 1986.

Olson, T.J., "An image processing package for the BBN Butterfly parallel processor, Butterfly Project Report 8, Computer Science Dept., Univ. Rochester, September 1986.

Olson, T.J., "Modula-2 on the BBN Butterfly parallel processor," Butterfly Project Report 4, Computer Science Dept., Univ. Rochester, January 1986.

Rigoutsos, I. and C.M. Brown, "Camera calibration," TR 186, Computer Science Dept., Univ. Rochester, to appear, 1986.

Sanchis, L.A., "Multiple-way network partitioning," TR 181, Computer Science Dept., Univ. Rochester, March 1986.

Scott, M.L., "The interface between distributed operating system and high-level programming language," Butterfly Project Report 6, Computer Science Dept., Univ. Rochester, revised September 1986; also, TR 182, Computer Science Dept., Univ. Rochester, revised September 1986; also, *Proceedings*, Parallel Processing Conf., October 1986.

Scott, M.L., LYNX Reference Manual," Butterfly Project Report 7, Computer Science Dept., Univ. Rochester, revised August 1986.

Scott, M.L. and A.L. Cox, "An empirical study of message-passing overhead," Butterfly Project Report 17, Computer Science Dept., Univ. Rochester, December 1986.

Sher, D.B., "Optimal likelihood generators for edge detection under Gaussian additive noise, "TR 185, Computer Science Dept., Univ. Rochester, June 1986; *IEEE Proc., Conf. on CVPR*, June 1986.

Sher, D.B., "Evidence combination for vision, using likelihood generators," *Proc., DARPA Image Understanding Workshop*, Miami, FL, December 1985.

Sher, D., "Template Matching on Parallel Architectures," TR156, Computer Science Dept., Univ. Rochester, July 1985.

Sher, D. and A. Tevanian, "A Hough Chip," internal course project report, Computer Science Dept., Univ. Rochester, April 1983.

Book Chapters:

Allen, J.F., "Maintaining Knowledge About Temporal Intervals," in *Readings in Knowledge Representation*. R. Brachman and H. Levesque (eds). Los Altos, CA: Morgan Kaufmann Publishers, Inc., 1985.

Allen, J.F., "Speech Acts," in *Encyclopedia of Artificial Intelligence*. S. Shapiro (ed). New York: John Wiley & Sons, Inc., in press, 1987.

Allen, J.F. and H.A. Kautz, "A Model of Naive Temporal Reasoning," in *Formal Theories of the Common Sense World (Vol. 1)*. J.R. Hobbs and R. Moore (eds). Norwood, NJ: Ablex Publishing Co., 1985.

Allen, J.F. and C.R. Perrault, "Analyzing Intention in Utterances," in *Readings in Natural Language Processing*. B. Grosz, B. Webber, and K. Sparck-Jones (eds). Los Altos, CA: Morgan Kaufmann Publishers, Inc., 1986.

Brown, C.M. (ed). *Advances in Computer Vision*. Hillsdale, NJ: Lawrence Erlbaum Associates, Pub., 1986.

Brown, C.M., "Space-Efficient Hough Transformation for Object Location," in *Statistical Image Processing and Graphics*. E. Wegman (ed). Marcel-Dekker, in press, 1987.

Feldman, J.A., "A Functional Model of Vision and Space," in *Vision, Brain and Cooperative Computation*. M. Arbib and A. Hanson (eds). Cambridge, MA: MIT Press/Bradford Books, in press, 1987.

Feldman, J.A., "Energy Methods in Connectionist Modelling," to appear, *Pattern Recognition, Theory and Applications*, P.A. Devijver (ed.), NATO ASI Series in Computer Science, Springer Verlag, 1987.

Feldman, J.A., "Massively Parallel Computational Models," in M. Commons (ed). Cambridge, MA: Harvard University Press, in press, 1987.

Feldman, J.A., "Neural Representation of Conceptual Knowledge," to appear, forthcoming book, L. Nadel *et al.* (eds.), 1987; also, TR 189, Computer Science Dept., Univ. Rochester, June 1986.

Feldman, J.A. and L. Shastri, "Evidential Reasoning in Semantic Networks," to appear, forthcoming book, Wilks and Partridge (eds.), 1987.

Kautz, H., "Formalizing Spatial Concepts and Spatial Language," in *Formal Theories of the Commonsense World*. J. Hobbs (ed). Stanford, CA: Center for the Study of Language and Information, 1985.

Shastri, L. and J.A. Feldman, "Neural Nets, Routines and Semantic Networks," in *Advances in Cognitive Science*. N. Sharkey (ed). Ellis Horwood Publishers, in press, 1987.

Image Understanding and Robotics Research at Columbia University

John R. Kender[1]
Peter K. Allen
Terrance E. Boult

Department of Computer Science
Columbia University, New York, NY 10027

0 Introduction

The Vision and Robotics Laboratory at Columbia has attained some measure of maturity. Three professors of computer science (John Kender, Peter Allen, and Terry Boult) and one research computer scientist (Hussein Ibrahim) lead a staff of two research programmers (Earl Smith, lab manager, and Lisa Brown) and a body of approximately 15 graduate students (listed below). Recent equipment grants have augmented our VAX, Sun, Grinnell, and Puma facility with a Masscomp real-time processor, and will shortly add an IBM arm and a real-time pyramid-based Aspex image processor.

Our research investigations reflect the principal interests of the four faculty members. Several of these investigations are joint projects with non-vision/robotics faculty members of the Department of Computer Science; such faculty are listed below, together with the students and staff associated with the effort. Briefly, from vision through parallel algorithms to robotics our interests are:

1. Low-level vision:
   - a. Generalized stereo, including spectral and polarization stereo, and surface curvature photometric stereo (Larry Wolff).
   - b. Approximation methods in stereo (Ken Roberts, with Dr. Sundaram Ganapathy of AT&T Bell Laboratories).
   - c. Tracking of non-rigid bodies (George Wolberg).
2. Middle-level vision:
   - a. Mathematical complexity, and psychological validity, of surface reconstruction and segmentation, with application to super-quadrics (Ari Gross).
   - b. Fusion of multiple shape-from-texture methods (Mark Moerdler).
   - c. Shape from darkness: a method for deriving surfaces from dynamic shadows (Michalis Hatzitheodorou and Earl Smith, with Prof. Grzegorz Wasilkowski).
3. Spatial relations:
   - a. Quantitative aspect graphs and maps (David Freudenstein and Marc Scott, with Prof. Jonathan Gross).
   - b. Disassembly representations and heuristics (Ken Roberts).
   - c. Landmark definition, and the representation and exploitation of cognitive maps (Matthew Blaze and Avraham Leff).
   - d. Natural language generation of spatial relationships, particularly via prepositions (Michal Blumenstyk, with Prof. Kathy McKeown).
4. Parallel Algorithms:
   - a. Basic image algorithms, and the integration of stereo with multiple texture algorithms (Lisa Brown).
   - b. Depth interpolation using optimal numerical analysis techniques (Dong Choi).
   - c. Basic image algorithms on the Aspex image processor (Ajit Singh).
5. Robotics:
   - a. The modeling and integration of multiple sensors, particularly vision and touch (Amy Morishima).
   - b. Geometric modeling and reasoning, with emphasis on partial descriptions and occlusions (Dino Tarabanis).
   - c. Interactive software environments (Yael Cycowicz, with Prof. Gail Kaiser).

We now detail these efforts, many of which are documented by papers in these proceedings. We also include short discussions of work in progress.

1 Low-level Vision

We have explored several issues in low-level vision, all of which arose from problems in dealing with real imagery, and all of which were related to our interaction with related research at AT&T Bell Laboratories.

[1]This work was supported in part by the Defense Advanced Research Projects Agency under contract N00039-84-C-0165.

1.1 Generalized Stereo: Spectral/Polarization

We have reexamined the physics and the assumptions underlying the phenomena of both binocular and photometric stereo, and have succeeded in generalizing them to a large class of methods for determining surface orientation, which we call "physical stereo" [21]. The name derives from the insight that virtually any measurable image quantity which has a determinable variation described by laws of physics can serve as the basis for the recovery of analytic and geometric surface properties, including depth, gradient, and curvature.

Differences in observed angle under unchanging illumination but changing viewpoint (binocular stereo) are related to surface depth, and differences in observed irradiance under changing illuminant position but unchanging viewpoint (photometric stereo) are related to surface orientation. However, both can be generalized to other observations: differences in irradiance under changing wavelength or polarization can be related to surface orientation, too, as can certain differences in the Fourier domain that arise under changing object position and rotation. Any observed changes due to variations in surface temperature or electrical current flow can be exploited also, given certain reflectance models. Even changes due to variations in the propagating medium ("mist stereo") are quantifiably useful.

We have first generalized and extended the method of photometric stereo to spectral stereo and polarization stereo, two related methods which can be used singly or in combination with each other [22]. Spectral and polarization stereo methods consider the intersection of equi-reflectance curves in gradient space, while varying the wavelength and/or the polarization of light emanating from a single light source as the imaging geometry is left invariant. In order to capture the observable effects of varying both of these physical properties, we replaced the ubiquitous Lambertian reflectance model with a more realistic one in which they appear as parameters: the Torrance-Sparrow model. The model is applicable to a wide variety of isotropically rough surfaces ranging from metals to paper.

An analytic study, backed by simulations of the error behavior, resulted in the following predictions on experimental accuracy which are in the process of being tested. Polarization stereo should work best with dielectrics (for example, magnesium oxide), which are virtually insensitive to spectral variations; roughly speaking, the situation is reversed for conductors (for example, aluminum). For polarization stereo, best results should be obtained when two separate observations are taken with light polarized parallel to the plane of incidence and perpendicular to it, respectively. As in photometric stereo, precision is paramount, since the methods are sensitive to error. We have discussed two methods for solving for the surface gradient: either directly by intersecting equi-reflectance curves for two different light qualities, or indirectly by using the two observations to decompose the reflectance function into its specular and diffuse components, which are themselves then intersected.

Unlike photometric stereo, a third independent equi-reflectance curve is impossible in spectral or polarization stereo, due to the inherent symmetry of isotropic reflectance functions. Thus, the method results in an intrinsic ambiguity in the surface orientation. However, by using a variable, asymmetric aperture over the single light source, we calculated that it should be possible to simulate the effect of multiple light sources in the manner of pure photometric stereo.

1.2 Generalized Stereo: Surface Curvature

In related work, we demonstrated how photometric stereo can be used to directly compute principle curvatures at each point on an arbitrary smooth surface [23]. This obviates the existing technique of explicitly differentiating the normal map derived from standard photometric stereo. Further, the method does not require auxiliary assumptions about the underlying surface. What is exploited instead is the image gradient: additional image information heretofore neglected.

We demonstrated that the availability of surface gradient and image gradient information allows the calculation of the local image Hessian, which gives complete knowledge about the second order rate of change of a smooth surface with respect to the image plane. We further derived that from the image Hessian it is straightforward to compute the curvature matrix for any visible point on the surface. Principle curvatures and their directions are given by the eigenvalues and eigenvectors of this matrix, and the Gaussian curvature is given by its determinant. If needed, lines of curvature are obtainable in the image by finding integral curves through the projected curvature vector field. An interesting observation we determined is that the sign of the determinant of the image Hessian is equal to the sign of the Gaussian curvature; thus, a qualitative (but crude) segmentation of the surface into curvature classes is possible directly from image observables.

Our current research in this area attempts to further explore physical stereo and its relation to differential geometry. Perhaps it will culminate in a unified paradigm for low-level vision.

1.3 Approximation Methods in Stereo

We succeeded, under limited assumptions, to quantify the errors that occur in the three-space measurements derived under binocular imaging. The result was reassuring: in its most simplified form it confirmed existing but previously unexamined practice.

When two corresponding pixels of a stereo pair of real images are back-projected into three space, they often do not intersect due to alignment or calibration error. The situation is exaggerated if the positions of the pixels themselves are uncertain. Then, the uncertainty of a position in the image plane translates into a cone of uncertainty in three space, and the intersection of the two cones is a badly-behaved volume of uncertainty in three space. What point in this volume is the most likely candidate for the originating stimulus in the images?

Through standard convolution techniques it is theoretically possible to determine the a priori probabilities of each such point in three space; however, in the general case, the actual convolution is intractable. Simplifying the problem somewhat, we assumed that the image observations suffer from position errors with a Gaussian distribution; the cones of uncertainty then have Gaussian cross sections. Although the convolution itself remains intractable, if one solves only for the space of most likely three-space locations, an exact closed-form solution is obtained. The locus is determined solely by that component of the image disparity vector that lies parallel to the camera baseline: it is as if the back-projected lines were truly coplanar to begin with. Thus the typical practice of ignoring off-baseline components of the disparity vector is justified, at least to the extent that the assumed Gaussian model is valid.

The method easily extends to multiple cameras or multiple views.

1.4 Non-Rigid Body Tracking

In work originally stimulated by an interest in character recognition [18, 20], we are investigating the problem of inexpensive motion tracking of nonrigid bodies over long series of (100 or more) frames. Our current technique assumes an initial segmentation of the image into regions that correspond to objects. To attack the fundamental problem of computational cost, we found it useful to discretize both the spatial and the intensity dimensions. The approach is then one of coarse-fine control, along these two orthogonal features.

Elements in the coarse spatial representations were first labeled with the label of the most prominent object covering it. In successive frames, the algorithm would look for the nearest neighboring label with similar intensity data. In cases of ambiguity, the algorithm had to search in neighboring regions for the proper correspondence; when found, the appropriate label would be propagated.

The method worked well even for objects that moved quickly and changed shape, as long as sufficiently high contrast was maintained: that is, as long their intensities were sufficiently different than those of nearby objects so that the coarse intensity data was a sufficient discriminator. In general, it also worked well for the interior of large regions. However, there was occasional deterioration at the common borders of regions with similar intensities. Without additional shape constraints or temporal processing, this appears unavoidable.

We are exploring additional, inexpensive representations that can be used to establish the best match in one frame from information derived from a short sequence of previous frames. This appears to involve a scale-space filtering of the objects that were initially segmented; further, the number of dimensions we coarsify will likely grow to three, to include the dimension of time.

Aside from obvious real-time applications, this work has direct impact on the automatic colorization of movies.

2 Middle-level Vision

We have found the middle levels of vision a rich ground. In addition to the mathematical analyses common to this level, we are able to report on the successful application of techniques of cognitive psychology to a difficult methodological problem, on the preliminary implementation of a middle-level system that fuses multiple conflicting knowledge sources, and on a new method for shape recovery that has industrial vision applications.

2.1 Surface Reconstruction: Optimality

We have explored the complexity of the task of reconstructing smooth surfaces from sparse depth data, approaching it in two novel ways. The first way is computationally, by using the tools and techniques of information-based complexity [3, 6]. After generalizing the traditional formulation of the problem to the two related problems of surface interpolation and surface approximation, we stated the optimal abstract mathematical solutions and discussed four realizations of them. Two of these four approaches are nontraditional and have been previously overlooked, but they permit the use of functions with fractional numbers of derivatives.

We compared all four approaches, calculated their computational complexity, pointed out ways to optimize performance, and detailed the image-taking circumstances under which each approach was most appropriate [4]. The results were extensively discussed, and documented with tables of timings too voluminous to summarize here. Within each of the approaches, we demonstrated that the exact mathematical solution depends critically on tunable parameters that define the meaning of "smooth". This led us to the conclusion that mathematics alone cannot provide powerful enough criteria for evaluating the success of a surface approximation method.

Our current research attempts to use the tools of information-based complexity again, this time to attack the problem of optimal surface segmentation.

2.2 Surface Reconstruction: Psychology

Our second novel approach to the complexity of surface reconstruction was psychological, by designing experiments to empirically determine what it was that human beings meant by "smooth". We devised two experiments: one to measure the detectability of discontinuities in one-dimensional curves, and one to rank the goodness of surface reconstruction under the methods investigated mathematically above [5, 7]. In the first experiment we established that human beings are blind to discontinuities in the second derivative unless they are large. The second experiment strongly suggested that for surfaces with discontinuities, the non-traditional approaches are considerably more accurate reconstructors, even though they perform independently of the value of the smoothness parameter.

2.3 Surface Reconstruction: Super-Quadrics

In a related effort, we have begun to investigate the utility of super-quadrics as a model for surfaces in image understanding. Although super-quadrics appear to be powerful tools for computer graphics, allowing complex solids and surfaces to be constructed and altered easily from a few interactive parameters, it is not immediately clear if these graphics features are advantageous for the purposes of surface reconstruction.

We have begun experiments to recover some or all of the eleven super-quadrics parameters directly from depth data. Noise in the data is unavoidable, and certain super-quadric parameters appear to be rather unstable when "run backwards". Thus, we have overconstrained the problem by using a pyramid-based relaxation scheme to quickly propagate local parameter estimates to neighbors.

2.4 Knowledge Source Fusion

Robust vision systems will almost certainly need multiple knowledge sources, possibly coordinated through a central blackboard [14]. We are investigating this issue by developing a system which attempts to integrate the output of several shape-from-texture methods [17].

We have devised a new data structure, the augmented texel, which in a compact notation records and resolves multiple orientation constraints for each image texel. Each such constraint arises from a shape-from-texture module, which conceptually runs autonomously and in parallel with several others. Using a Hough transform-like method on a trixelated Gaussian sphere, the system estimates the most likely orientation for each texel. The system can then segment the image by inferring which patches share the same surface orientation; the inferencing scheme can even separate two superimposed (mostly) transparent surfaces.

We have demonstrated the approach using the knowledge sources of shape from uniform texel spacing and shape from uniform texel size, on real imagery of both controlled and natural textures. Accuracy appears to be within five degrees or less. The system is being augmented with additional knowledge sources, and we intend to explore other control structures beside the augmented texel.

2.5 Shape from Darkness

We have developed a new, active method for extracting surface shape information from the information implicit in dynamic shadows and self-shadows [15]. This method has two major advantages over previous work. First, it appears to work best for textured objects: where existing methods fail most badly. Secondly, it is more robust than existing methods in that it requires no a priori information about a surface's reflectance or its smoothness.

The method gathers shadow information in a novel image representation called the suntrace, which records the "time of day" (illumination angle) at which each pixel first becomes illuminated. Using a relaxation scheme to refine upper and lower bounds on the surface, the method solves the underlying set of overdetermined equations very rapidly and accurately. Convergence in two-dimensions is aided by the existence of eight constraint equations for each pixel's upper or lower bound, since each pixel can (and usually does) shadow four others, and is itself shadowed by still four more, at differing times. Many results have been simulated, and research using already collected real shadow data is continuing.

We have analyzed the problem with the methods of information-based complexity, and have succeeded in finding a near-optimal algorithm for the problem restated in continuous mathematics. The solution is an interpolating spline of a peculiar sort, which nevertheless is highly accurate even for small amounts of data. However, the true optimum remains subject to investigation, as does the related question of the optimal number and placement of light sources. The inaccuracy of most real edge detectors demands that the error behavior of the algorithm be studied, too.

The method should have direct application as a new type of non-contact robotic sensing, most probably within an industrial setting. A preliminary patent search has confirmed the novelty of the approach.

3 Spatial Relations

This is our newest effort. Most of the work is still in progress. We do not view it as "high-level vision" because the emphasis is not solely on single objects and their models. It is more concerned with vision in the very large, where even perhaps sight is not a necessary input to the reasoning processes we are investigating.

3.1 Aspect Graphs

An aspect graph is a representation of the dependency of an object's observable features on the direction of view. Usually the observable feature is topology of shape, outline, or line drawing; the aspect graph records the number and interrelation of the equivalence classes that arise over the entire viewing sphere. While attempting to extend the concept of aspect graph of objects to the concept of aspect graphs of object groupings, we discovered several inadequacies in its current definition, some errors in several derivations, and some problems with its current use [16].

We demonstrated that the concept of "characteristic view" is not well defined; the definition can be salvaged only in terms of the total performance of a recognition system. We also displayed corrected aspect graphs. Lastly, since no camera is ideal, the grounding of an aspect graph in topology is unrealistic and must be extended to include some dependency on camera resolution: an aspect "map". We therefore quantified, in some simple cases, how likely a characteristic view would be given a camera specification and an object size. Thus, as camera resolution decreases, some aspects actually become impossible.

Present work has returned to the problem of object formations, with special emphasis on the problem of automatic derivation of their aspect maps from the aspect maps of their components. The ultimate goal is the automatic deduction of camera placement strategies that maximize sensed information while minimizing sensing cost.

3.2 Disassembly

The intelligent disassembly of unknown objects appears to be a novel and challenging way to investigate the efficiency and predictive power of representations for spatial relationships. Further, such a task would rigorously exercise the control structures involved, since the openness of the domain appears to require planning and search of an intrinsically heuristic kind. Disassembly may even provide a basis for experiments in learning.

We have begun to explore the cognitive economy of several representation schemes, using a simulated, quantized, two-dimensional universe of simple objects. Our simulated robotic agent will disassemble a new object by the repetition of the following (or related) steps. First, having operationalized advice such as "no component is longer than three units", it will reduce the problem if possible, exiting if complete. Then, it should plan the most efficient single action, using heuristics. Lastly, it will observe and compare the results to what was anticipated, and deduce implicit relationships from apparent ones: for example, parts that maintain the same spatial relationship to another part that has been moved are probably physically attached to it.

Given the appropriate selection of real world objects, it should not be difficult to demonstrate such a system with a camera and robot arm.

3.3 Landmarks

We have noticed an analogy between following a map during the course of navigation, and following a script while understanding natural language. In both cases stereotyped events serve as cues to positions in a topological abstraction of real world activity; the events in navigation are called landmarks. Both maps and scripts reduce wandering, without entirely eliminating choices of processing direction. We intend to pursue this analogy further, and to define and resolve several assumptions unjustified in script theory: for example, what makes an event stereotyped?

We also wish to build a system that will demonstrate the answers to questions of the following sort. How can maps be acquired and refined? Where am I, and how do I know I am or am not lost? How can I efficiently track my map-generated symbolic predictions to my sensor-based signal observations? How can experience be effectively compressed into a symbolic map? How do I most effectively report on my own experiences?

We have developed a small LISP system that models landmark definition and acquisition. It monitors a simulated robot coursing thorough a one-dimensional(!) world. Having been given rudimentary models of sensor, memory, and actuator performance, it compresses its "experience" into a retrievable, compact, fuzzy form. Figuratively and literally, we still have far to go.

3.4 Natural Language Generation

We are investigating how natural language can be used to describe the contents of a scene. We have discovered that the entire area of descriptive spatial relations, even within the discipline of computational linguistics, is nearly untouched. In particular, the pathway between spatial relationship and their realization in surface language constructions--mostly via prepositional phrases--is very inadequately known.

We are building a system which takes as input a simple two-dimensional scene, and attempts to answer questions of the form "Where is X?". Answering is not simple, and requires at least two intermediate steps. First, the system must decide on a referent object Y whose identity and position are known to the questioner: this is a standard concern of natural language algorithms that track context, focus, history, and function. Secondly, the "deep" spatial relationship of X to Y must be determined; unfortunately, no catalog of such relations exists as yet. Once X is deeply related to Y, the surface construction should be relatively easy to construct, and, in fact, a system to do so is already operating in a different setting. We therefore are concentrating on accumulating empirical evidence of people's use of prepositions, and attempting to obtain a network structure, perhaps hierarchic, to represent it.

4 Parallel Algorithms

We have continued our investigations into fine-grained SIMD machines, partly under our basic research contract, and partly under the Strategic Computing effort. Our concentration has been on the middle-levels of vision, where it appears that most algorithms are not straightforwardly pixel-based. Successful extraction of superior performance therefore requires some care in algorithm selection and analysis, and in the decomposition of both data and control.

4.1 Fine-Grained SIMD: Basic

We have adapted many low-level vision algorithms to fine-grained tree-structured SIMD machines (basically, pyramid machines), and have discovered several novel approaches to programming them that avoids the bottlenecks associated with the high data rate through the root of the tree (or the apex of the pyramid). Through vertical pipelining, data duplication, and subproblem partitioning we were able to demonstrate the achievement of massive parallelism [10, 11, 13]. We demonstrated the results in three ways: analytically, through simulation, and by actual execution on real data on a prototype 63-node machine (NON-VON I). Our results compared favorably with other proposed or existing architectures, although algorithms consisting of repeated nearest neighbor operations (for example, convolution) suggested that tree connections ought to be complemented with mesh connections. Otherwise, traffic through the root degrades performance, despite novel ways of decomposing and redundantly prestoring the image into subtrees, and despite provably optimal algorithms for image shifting.

At the middle level of image understanding, we were able to demonstrate algorithm decomposition and parallel speed-ups of two middle-level tasks: the Hough transform, and the matching portion of an algorithm to interpret moving light displays [12]. These, too, were analyzed and simulated. Since this work was apparently the first to consider these middle-level problems, it was not possible to compare our results with those of other architectures. (Since publication, several researchers have implemented the Hough transform on other parallel computers.) High performance was attributable to the careful duplication of image data and/or control decisions throughout the tree, and by the deliberate avoidance of communication of intermediate results. In short, we found that for middle-level tasks it was necessary to program the fine-grained SIMD architecture as if it were a type of moderate-grained Multiple-SIMD (MSIMD) machine.

4.2 Fine-Grained SIMD: Stereo with Texture

Our present and future work on parallel architectures is directed at the specific problems of stereo, texture, and their cooperative fusion. We have implemented on our simulator a multi-resolution stereo algorithm based on a del-squared-G edge detector. This work included the implementation of more than one algorithm for image convolution on tree machines, and an exploration for the appropriate parallel algorithm for establishing the direction (rather than the mere existence) of detected image edges.

We have analyzed, spatially decomposed, and simulated a parallel version of the Witkin statistical shape-from texture algorithm; additionally, we evaluated which portions were best executed on the high-speed attached supercomputer, and which in the conventional host. In the course of the analysis we discovered that the algorithm's accuracy is substantially improved if surface orientation is represented and measured on the surface of the Gaussian sphere, rather than in a slant-tilt space [8]. Further, we established that the most advantageous parallel decomposition of the problem associates a processing element with each possible surface orientation. Since the probability of each orientation is computed in parallel (and not just the single most likely one), this represtation can be exploited by the host processor to adjust various edge-detection parameters, or to compare this method's results with those from other shape-from methods.

As part of the DARPA sponsored workshop on parallel image understanding architectures, we calculated the performance of a representative set of image understanding parallel algorithms on fine-grained tree machines, with satisfying results.

Necessary infrastructure work includes the implementation on NON-VON of a subset of Connection Machine Lisp, and the continuing development of multiresolution structures and algorithms on our simulator testbed, the Grinnell image processor.

4.3 Depth Interpolation

We have investigated how to exploit fine-grained mesh-connected SIMD architectures for middle-level vision tasks such as depth interpolation. Significant computational gains are to be had by adopting algorithms of numerical analysis that are more sophisticated than the standard Gauss-Seidel approach [9]. What is necessary is that the architecture incorporate a way of quickly gathering information that is global to the entire image: for example, a tree, pyramid, or omega network should be superimposed over the mesh. With the pure parallelism of local mesh connections, and the logarithmic time parallelism of the global tree communications, the methods of adaptive Chebyshev acceleration or conjugate gradient enable qualitative speed-ups. These methods sit naturally and well on such machines, too.

We analyzed the time and space requirements for the solution of the singular positive definite matrices that arise in the depth interpolation problem (and other related middle-level vision problems, for example, shape from shading). Further, we analyzed and quantified the communication costs. We simulated the parallel solution of depth interpolation on images of up to 128 pixels square for several methods. We noted in all cases that conjugate gradient was the superior method, but that adaptive Chebyshev does indeed adaptively accelerate. As expected, the sparser the depth data, the more difficult the interpolation.

Since these initial experiments used only simple synthetic data, more elaborate synthetic images will be pursued in future. Further, we plan to run our algorithms on real images from binocular front ends or from directly sensed range data.

4.4 Image Processor Algorithms

Aided by our experience in parallel supercomputer design [19], we have begun to implement several existing parallel vision algorithms on the commercially available Aspex image processor, in support of various robotic tasks.

5 Robotics

Our final area of interest is our most systems-oriented. We are attempting to apply programming language techniques, as well as artificial intelligence ones, for the sake of more responsive, more autonomous robotic agents.

5.1 Models for Multiple Sensors

Multiple sensing is in many ways a problem in the design of distributed computation. Each sensor is typically a separate computing element, with its own world model, set of primitives, and physical characteristics. Extending any existing robotic system to incorporate a qualitatively different new sensor is strenuous work. Building an integrated manipulator out of the multiple components of many-fingered tactile sensing, force/torque wrist sensing, and real-time vision is truly formidable.

We are pursuing the application of the programming disciplines inherent in object-oriented languages to model generic sensor and effector properties [2]. The formalisms of classes and methods appear to promise a powerful but flexible framework for gaining the virtues of abstract sensor behavior without losing exact sensor control. For example, most tactile sensors provide relatively dense depth data (as opposed to vision, especially on uniformly colored surfaces). Since matches to database models are critically dependent on data density, the use of object classes should allow the abstraction of this common feature of tactile sensing into one method. The effectiveness of such an approach can be measured by the ease with which we can modify a robotic system based on a single sensing finger to one based on a sensing parallel jaw; the upgrade should occasion only a relatively minor amount of recoding.

5.2 Geometric Modeling and Reasoning

CAD/CAM systems can create many useful 3D models; however, they are usually volumetric-based. Thus, these models are not very useful for recognition, because sensors compute surface information: they cannot directly sense volumes. We have begun to investigate the integration of a existing, successful surface-based modeler based on Coons' patches [1] with a commercial CAD/CAM system. The existing system has demonstrated the utility of having flexibly intelligent means to trace and probe surfaces, cavities, and holes, and to efficiently match surface-based features to a surface patch database.

Further, we are attempting to extend the object descriptions to include functional as well geometric and topological attributes of objects. This will allow CAD/CAM systems to be used not only in design, but also in recognition, and even in some forms of intelligent reasoning tasks, such as the automatic substitution of parts, or the derivation of simple analyses of mechanical failures. We anticipate that rich object descriptions will make certain difficult robotic inference tasks easier: for example, the determination of a object from a partial or occluded view or touch, or the planning of effective strategies to efficiently determine the location and identity of totally obscured parts.

5.3 Environments

In a new effort, we are involved in the generation of an interactive environment for robotics programming in AML/X, a language developed at IBM Yorktown. We intend to use it for programming the soon to be delivered IBM robot arm. We are leveraging our development effort by modifying the SMILE programming environment, which is itself generated semiautomatically through the GANDALF environment generator. Among the tools under construction is a syntax-directed editor which hosts a language interpreter. We expect to add more application-specific aids as well.

6 References

1. Allen, P.K. Sensing and Describing 3-D Structure. Proceedings of the IEEE Conference on Robotics and Automation, April, 1986.

2. Allen, P.K. A Framework for Implementing Multi-Sensor Robotic Tasks. Proceedings of the DARPA Image Understanding Workshop, February, 1987. (These proceedings.).

3. Boult, Terrance E. Some Examples of Information-Based Complexity. Proceedings of the Bulgarian Academy of Sciences International Symposium on Optimal Algorithms, April, 1986.

4. Boult, T.E., and Kender, J.R. Visual Surface Reconstruction Using Sparse Depth Data. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, June, 1986.

5. Boult, T.E. *Information Based Complexity in Non-Linear Equations and Computer Vision*. Ph.D. Th., Department of Computer Science, Columbia University, September 1986.

6. Boult, T.E. "Optimal Algorithms: Tools for Mathematical Modeling". *Journal of Complexity* (To appear 1987), .

7. Boult, T.E. Using Optimal Algorithms to Test Model Assumptions In Computer Vision. Proceedings of the DARPA Image Understanding Workshop, February, 1987. (These proceedings.).

8. Brown, L.G., and Ibrahim, H.A.H. A Parallel Implentation and Exploration of Witkin's Shape from Texture Method. Proceedings of the DARPA Image Understanding Workshop, February, 1987. (These proceedings.).

9. Choi, D. Solving the Depth Interpolation Problem on a Fine Grained, Mesh- and Tree-Connected SIMD Machine. Proceedings of the DARPA Image Understanding Workshop, February, 1987. (These proceedings.).

10. Ibrahim, H.A.H., Kender, J.R., and Shaw, D.E. On the Performance of Tree-Structured Machines for Image Correlation. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, June, 1986.

11. Ibrahim, H.A.H., Kender, J.R., and Shaw, D.E. SIMD Tree Algorithms for Image Correlation. Proceedings of the National Conference on Artificial Intelligence, August, 1986.

12. Ibrahim, H.A.H., Kender, J.R., and Shaw, D.E. "On the Application of Massively Parallel SIMD Tree Machines to Certain Intermediate-Level Vision Tasks". *Computer Vision, Graphics, and Image Processing 36* (October 1986), 53-75.

13. Ibrahim, H.A.H., Kender, J.R., and Shaw, D.E. "Low-level Image Understanding Algorithms on Fine-Grained Tree-Structured SIMD Machines". *Journal of Parallel and Distributed Processing* (To appear 1987).

14. Kender, J.R. "Expert Vision Systems Demand Challenging Expert Interactions". *Computer Vision, Graphics, and Image Processing 34*, 1 (April 1986), 102-104.

15. Kender, J.R., and Smith, E.M. Shape from Darkness: Deriving Surface Information from Dynamic Shadows. Proceedings of the National Conference on Artificial Intelligence, August, 1986, pp. 664-669. (Also these proceedings.).

16. Kender, J.R., and Freudenstein, D.G. What is a Degenerate View? Proceedings of the NSF International Workshop on Geometric Reasoning, July, 1986. (Also these proceedings.).

17. Moerdler, M., and Kender, J.R. An Integrated System that Unifies Mulitple Shape form Texture Algorithms. Proceedings of the DARPA Image Understanding Workshop, February, 1987. (These proceedings.).

18. Pavlidis, T., and Wolberg, G. An Algorithm for the Segmentation of Bilevel Images. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, June, 1986.

19. Singh, A., and Lala, P.K. "A Multilayer Cellular Architecture for a Highly Parallel VLSI Supercomputer". *IEEE Transactions on Computers* (To appear 1987).

20. Wolberg, G. A Syntactic Omni-font Character Recognition System. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, June, 1986.

21. Wolff, L.B. Physical Stereo For Combined Specular and Diffuse Reflection. Department of Computer Science, Columbia University, November, 1986.

22. Wolff, L.B. Spectral and Polarization Stereo Methods Using A Single Light Source. Proceedings of the DARPA Image Understanding Workshop, February, 1987. (These proceedings.).

23. Wolff, L.B. Surface Curvature and Contour From Photometric Stereo. Proceedings of the DARPA Image Understanding Workshop, February, 1987. (These proceedings.).

Developments in Knowledge-Based Vision for Obstacle Detection and Avoidance†

K.E. Olin, F.M. Vilnrotter, M.J. Daily and K. Reiser

Hughes Research Laboratories
Artificial Intelligence Center

Abstract. In this collection of Image Understanding Workshop Proceedings, the applications of image understanding into the domains of autonomous vehicles, cartography, target recognition, and robotics are being emphasized. Progress at Hughes Artifical Intelligence Center as it relates to these applications is summarized in this overview.

Hughes is concerned with the detection and avoidance of obstacles for an autonomous land vehicle. We concentrated our early efforts on road scenarios, but more recently have been researching issues related to cross country manuevers. The problem of navigation for an autonomous land vehicle requires the integration of systems for perception and planning. We have developed a system architecture which supports multiple levels of assimilation and immediacy required for perceptual reasoning as well as the integration with path planning. The architecture exploits new concepts of perceptual *virtual sensors* in conjunction with reflexive behaviors for path planning. This architecture is summarized in this overview together with specific algorithm developments exploiting color imagery, sensor fusion, and the modeling of a symbolic local map. A separate paper describes the laser range sensor processing procedures we have developed for obstacle detection. This work was accomplished through the team effort of the authors together with J.G. Harris.

It is through many years of experience with automatic target detection systems that the concepts of context dependent cueing evolved. A system has been developed which uses scene context to aid in target detection.†† The scene context currently exploited includes target motion, spatial relationships between targets (e.g. formation knowledge), and spatial relationships between target and background (e.g. target/road knowledge). This system will be featured in an invited technical presentation and report entitled "Image Interpretation Using Scene Context" by T.M. Silberberg.

In addition, Hughes has been pursueing research in multiprocessor architectures for vision applications. We have designed and implemented our own machine, the Hierarchical Bus Architecture, with which we are currently performing experiments with low and mid level vision algorithms. This work, performed by R.S. Wallace, will be summarized in this overview.

1. Autonomous Land Vehicle

1.1 System Overview

Navigation for the Autonomous Land Vehicle (ALV) is a complex problem requiring the integration of perception and planning systems. We have defined an overall system architecture which supports a decomposition of the problem directed by immediacy and assimilation considerations as shown in Figure 1. The vertical structure generally reflects the level of information fusion; the highest level application for mission knowledge assimilates symbolic information for the longest period of time and the largest spatial area, while the lowest level application requires information at very rapid update rates. For instance, an application at the highest level may necessitate the recognition of a sequence of landmarks to be used for navigation, while an application at the lowest level may simply be the tracking of the left or right road edge at near real-time video data rates. Within each level there is a horizontal structure to provide the flexibility and variability needed in the ALV problem domain. The horizontal structure may be thought of as groups of sub-experts, each contributing the knowledge and processing needed to satisfy specific information requests. With this decomposition, new modules may be easily added to the system as the ALV capabilities develop. The majority of our technical efforts to date have been in the horizontal decomposition of the lower level modules. This work is discussed in this overview and in the technical paper describing Hughes laser range processing. A reference describing the parallel efforts at Hughes in path planning is found in [1].

Perception is tasked with providing the planner with information about the currently perceived environment. Presently, perception is defined as the ability to "see" the environment with color and laser range sensors, but it is conceptually expandable to include sensors that measure such things as temperature, water velocity, and soil viscosity. Perception needs to provide this information in a timely fashion, avoiding the use of outdated information for path planning. Perception functions in response to requests from the planner. These requests are for information with a specified accuracy and update rate. Each level of the planning system makes requests of perception at the corresponding level. Interpretation of a request may send subtasks down the perception hierarchy. Therefore, the perception system is internally communicating information up and down levels, as well as externally communicating to the planner from any level. It is perception's job to translate these requests into sensor commands and a set of processes which will provide the information needed to satisfy all requests. We expect that perception will need to communicate at both high and low data rates and with multiple accuracy requirements. This means that perception may not have the processing time available to assimilate all information completely. Instead, information requiring limited processing and associated with lower accuracy or confidence may be made available to the planner at faster rates. If the planner requires greater assimilation of perception information, slower data rates may result.

† The research described in this paper has been supported by Defense Advanced Research Projects Agency (DARPA) contract #DACA76-85-0007 unless declared otherwise in the text.

†† Development of this system has been supported by Night Vision and Electro-Optics Laboratory Contract (NV&EOL) #DAAK21-84-C-0131.

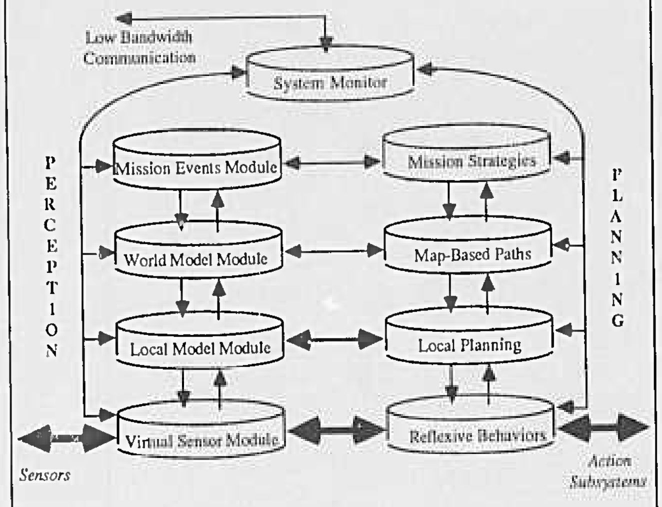


Figure 1. System Architecture

Activation of scene modeling within the perception system is restricted to those areas which have actually been sensed. The physical size and update rate of a model may vary depending upon the mission and terrain type. At the higher more global levels, we expect the world model to be on the order of hundreds of meters in size and updated on the order of tens of seconds. The local model size is optimized for the vehicle size and speed and the field of views (FOV) of the sensors currently activated. Typically, the size is on the order of tens of meters and the update rate is on the order of seconds. At the lowest level of the hierarchy are the virtual sensors, which have the highest requirement for immediacy and in many cases the lowest degree of knowledge assimilation. Virtual sensors must have response times in the millisecond range and may reflect only a small portion of a sensor FOV. A virtual sensor may be thought of as an optimized procedure that supplies the highly coordinated information necessary to satisfy a reflexive action.

Within the planning community there is much discussion as to whether temporal tracking of an obstacle is necessary for avoidance. One argument is that for local planning the vehicle needs to respond quickly to the environment, which is highly dynamic at this level. Therefore, the vehicle should always use the most recent information available and plan as best it can until the next information update. The lowest level of our architecture using behaviors and virtual sensors is an example of this kind of navigation. A second approach to navigation minimizes local replanning and abrupt changes in vehicle control functions through incremental information updating. Tracking an obstacle for updated location supports this approach to navigation. Matching object features for tracking is performed at our local map level. Our architecture will support either or a combination of approaches.

We have begun implementation of our system at the lowest level where there is the greatest need for immediacy and the perceptual and planning efforts are most closely coupled. As a result of this work numerous algorithms have been developed for obstacle detection using laser range and color data. Many of these algorithms are now being evaluated as virtual sensors and implementation with corresponding reflexive behaviors. Testing the integration of virtual sensors and behaviors has recently begun in a simulated environment and is discussed in the evaluation section. The concept of virtual sensors will now be described.

1.2 Virtual Sensors

At the lowest level of the planning system is the Reflexive Behavior Module. This module is critical to real-time control because it provides a means to isolate control loops from other planning tasks. Reflexive behaviors are procedural units with high demands for immediacy. Typical obstacle related behaviors might be "slow-for-obstacle" or "turn-for-obstacle." Virtual sensors are conceptual entities which provide a symbolic means to couple perceptual information with reflexive behaviors. A virtual sensor is a blending of raw sensor data with specialized processing in response to an established contract between the perception and planning systems. This contract specifies what information is requested, the rate at which the information is to be provided, the format of the data (if multiple data types are available), the accuracy of the information, and the priority of the request. It is important to note that the virtual sensor contract is symbolic; that is, specific information, not algorithms, are requested. This provides an important communication channel between the perception and planning systems: the planner has the means to make symbolic requests, and perception has the means to translate the requests into sensor and processing commands. This provides the best opportunity for perception to satisfy the contract constraints or notify the planner of failure.

The relationships between reflexive behaviors, virtual sensors, image processing, and physical sensors is shown in Figure 2. Through this configuration, the planner can think of perception as a group of black boxes, each of which senses particular scene objects. Virtual sensor processing consists of low level feature extraction together with specialized procedures to obtain the requested information. For example, the right-road-edge virtual sensor implemented by Hughes classifies color features to locate road pixels and then applies specific search and link procedures to obtain the road edge. However, other road finding algorithms [2,3,4] could also be represented as virtual sensors to satisfy the same goal. This redundancy is a very attractive quality of the Virtual Sensor Module (VSM). Different approaches requiring different physical sensors, processing time and resources, and processing accuracy will allow the vehicle to function under a wider variety of conditions. The actual algorithm chosen must satisfy the contractual requirements of the current virtual sensor request. Ancillary knowledge associated with scenario, processing environment and mission goals may be used as an aid in algorithm selection.

Virtual sensors provide a methodology for responding to multiple tasking within an asynchronous environment. Each virtual sensor establishes an independent contract for throughput, data, and timing. Virtual sensors are conceptual objects with attribute slots being shared by the VSM and the behaviors. The object is associated with a time flag through which the reflexive behavior can check for new data from the virtual sensor. Most often virtual sensors provide very specialized information in order to satisfy the requirements for immediacy. This information is typically obtained in an algorithmic fashion without using complex reasoning strategies. Therefore, errors in accuracy will result, but because of the fast update rates recovery from these errors should have minimum impact on navigation. In addition, the Local Map Module is performing in a parallel fashion the higher level functions of sensor fusion, temporal reasoning, and spatial reasoning thereby forming a more complete scene model. With information from the local map, behavioral inconsistencies are determined by the planner. For instance, the virtual sensor request for finding the right side of the road may have different constraints than the virtual sensor request for the left side of the road. This reflects a real scenario in which the right side abuts the side of a mountain, such that

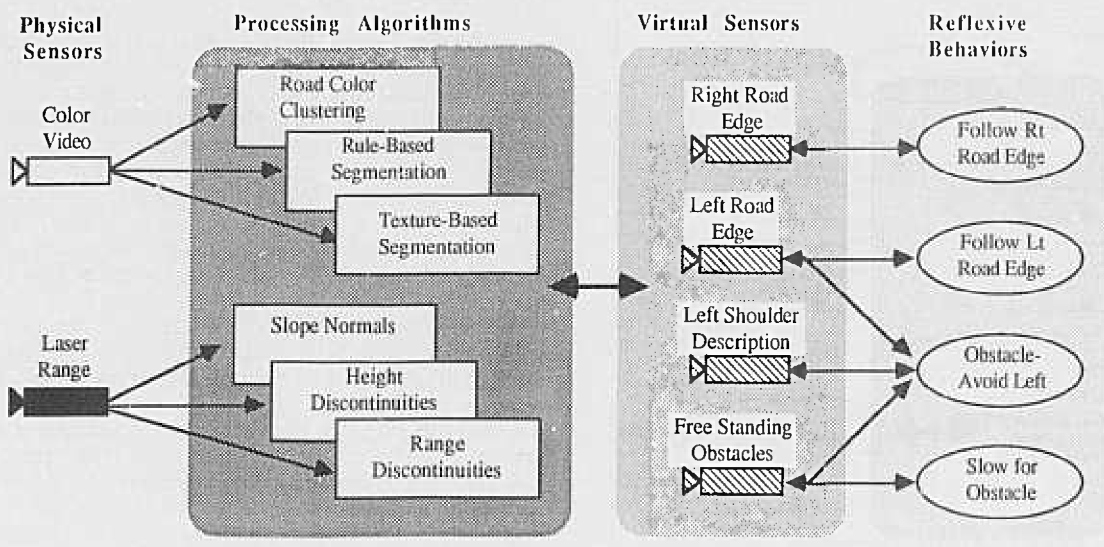


Figure 2. Virtual Sensor Approach

very accurate updates are required to avoid collision. In this case, the right-road-edge virtual sensor would use range data to accurately model the mountain side with laser range update rates of two frames/second. In comparison, the left-road-edge virtual sensor would continue to use a color classification at perhaps three frames/second or higher rates, but with a higher possibility of occasionally detecting erroneous road edges. It is conceivable that the left edge could produce conflicting information with the right edge. This is handled in two ways. First, the planner prioritizes and integrates the asynchronous information into a consistent set of vehicle control commands such that in this case the mountainous right side of the road will dominate the calculation of vehicle heading. This ability has already been demonstrated by the Hughes planner. Second, at the local map level of perception the conflicting information is flagged and a failure message will be sent to the planner. The planner may then choose to slow down until the road edge is regained or a new virtual sensor is initialized.

Reflexive behaviors which operate together are grouped into an *activity*. Within an activity, the relative behavior priorities, parameter constraints, and initialization and termination actions are assigned. When an activity is initialized, each component behavior is invoked as an independent process. More importantly, virtual sensors that feed the behaviors are initialized and data links are established between them. For efficiency, the virtual sensor configuration for each of the possible activities for a mission are initialized in advanced. When a new activity is invoked, any existing active behaviors and virtual sensors are disabled, the data links are released, and the new activation set is switched into operation. Activities may also be configured into sub-activity sequences for simple script execution. During initialization, sub-activities take into account future virtual sensor and data link requirements. This reduces switching time and minimizes the interruption of reflexive data links.

The virtual sensor control and arbitration system is being developed under Hughes Internal Research and Development (IR&D) funds. We are currently modeling the set of virtual sensors as a directed graph with the nodes representing the attributes of the virtual sensor and the arcs representing the input/output configuration. The virtual sensor attributes are in a frame representation and include processor and sensor knowledge. The control system is being modeled as a process scheduling problem to satisfy time, accuracy, and confidence constraints. As a result of the process scheduling, a process subgraph of the modeled graph will be selected for a requested group of virtual sensors. This process graph will be used to determine the required task and processor configuration.

1.3 Laser Range Analysis

The ALV is equipped with a laser range scanner which measures the distance along the line of sight to the nearest object. Distance (or "range") is actually computed by measuring at a specified angular interval the phase shift between the active laser signal and the reflected signal. This sensor inherently supplies information of surface geometries; however, the interpretation of this information is difficult for the complex outdoor terrain imagery associated with the ALV. We have developed several techniques for locating traversable ground, free standing obstacles, and terrain slope using range imagery. These techniques may be divided into two major categories: those which operate in the range image plane and those which operate in a Cartesian elevation map. Each method may also be applied in specific cases as a virtual sensor, returning only the required information about certain objects. For a complete description of our range imagery work, see the paper "Detecting Obstacles in Range Imagery" in these proceedings.

1.4 Color Analysis

Since our primary focus is to locate obstacles at very rapid update rates, our investigations using color imagery emphasizes techniques to extract information in support of object labeling. Therefore, our efforts have been expended in the development of simple color segmentation procedures. We have experimented in several color spaces with classical and rule-based classification schemes. In addition, a texture-based segmentation algorithm developed at Hughes has been employed. Virtual sensors have been developed to locate left and right road edges where there is no range discontinuity to demark a road edge.

1.4.1 Color Components. In natural scenes, most of the colors have very low saturation and occasional man-made objects or unusual regions have higher saturation. Color constancy is a problem for any classification scheme. For example, an image leading into the ALV/MMC test track corner taken on an overcast day in the summer was found to have a purple hue (top of Figure 4.) An image taken on the same day of the same corner only slightly further into the turn had a slightly more blue hue. Finally, an image of the same corner going in the opposite direction with full sun had a cyan hue (bottom of Figure 4.) This and other evidence suggests that similarly colored surfaces do not exhibit color constancy due to uncertain causes, such as camera settings, ambient lighting from reflectance or scattering, and perhaps seasonal changes (even for asphalt roads.)

The color video data available from the ALV data base is digitized in the RGB (red, green, blue) color space. The top row of Figure 3 shows the red, green and blue planes from the overcast image of the curve at the MMC test track. The color image data we have been working with to date has no reference color for maintaining constant values for a particular surface. Camera settings such as brightness, contrast and auto iris settings, color ratios for white or black surfaces, etc. have not been available; therefore, images of the same area digitized at different times may look very different. To compensate, we first rescale the red, green, and blue planes independently ensuring that the darkest pixel is really black and the brightest pixel is really white. This process does not maintain the color ratios of red to green to blue found in the original data, but our suspicion is that those ratios are not correct anyway since all of our data is skewed to the red.

The RGB color space does not separate intensity from each of the red, green, and blue pixel values making it subject to color constancy problems due to low sun angles, clouds, time of year, or other fluctuations in color over the same surface. This led us to explore one, two and three dimensional histograms of alternative color spaces, including IHS (intensity, hue, saturation), normalized color, I1I2I3 [5], opponent color, and YIQ. Our preliminary examination of the features associated with complex, natural terrain images indicates that for the data we have, the IHS space is as good as any other in discriminant power and still has human significance since it is possible to discuss the color of a region, the richness of the color, or the intensity of a region. While in the IHS space singularities exist where the RGB data has low intensity, potentially incorrect hue and saturation values can be ignored by checking for dark regions in the intensity component (where very little information is available anyway.) The bottom row of Figure 3 has the resulting intensity, hue, and saturation planes for the RGB image on the top row.

1.4.2 Classification Techniques. In order to focus only on the color of a pixel, we ignore the intensity component in the IHS model and use only hue and saturation within the framework of the statistical Bayes classifier. Mahalanobi's distance measurement was used to classify into *a priori* classes defined by mean and covariance. Hue and saturation are typically calculated in a polar coordinate system where hue is measured as an angle and saturation ranges from zero to one, zero meaning no saturation (i.e. white, black, or gray) and one total saturation. Since hue is an angle from 0 to 2π, and continuity around red is lost, the polar coordinate system is not appropriate for use with the Bayes method. By transforming into 2D Cartesian (X-Y space), hue and saturation values will cluster without the angular problems inherent in the polar coordinate scheme. Working with only hue and saturation in this representation, we have been able to distinguish colors of low saturation of natural objects differently from the highly saturated colors associated with man-made objects and shadows. We also implemented a Karhunen-Loeve (K-L) transformation which computes the optimal orthogonal feature vectors from the original data. The K-L transformation appears to have usefulness in segmenting images, and we will continue to develop methods which can adapt to more difficult scenes and use these features to improve segmentation. Figure 4 shows two scenes described previously which have been segmented using the Bayes classifier. The four classes in this segmentation are sunny road, cloudy road, dirt, and vegetation. The top segmentation consists mostly of cloudy road and vegetation classes with a few dirt pixels in the background. However, the segmentation of the road is disappointing. The segmentation on the bottom of Figure 4 does a much better job of classifying sunny road, dirt, and vegetation. The human observer with the RGB image can not distinguish the "dirt" region representing the road at a greater distance from the color of the dirt shoulders.

1.4.3 Rule-based Segmentation. One of our goals in both obstacle detection and the more general problem of scene description is to use as much of the available sensory information as possible. In this way, perturbations in the data such as those described above will have less effect on the overall description of the object. To correctly describe and label objects in a scene, we are implementing a rule-based segmentation scheme which uses as much information as possible from both the raw images and virtual sensors. In the segmentation shown in Figure 5, a coarse-grained description of a pixel is obtained by quantizing the IHS values into three levels of intensity, twelve levels of hue, and three levels of saturation. Preliminary results indicate that this type of quantization yields some degree of invariance to minor changes in the color of the same or similar surfaces. For example, after rescaling the RGB values as described above, asphalt roads almost always have one of the following hues: cyan-blue, blue, blue-magenta, or magenta. Furthermore, they tend to have very low saturation (i.e. less than 10%), and are typically high in intensity (greater than 170 on a scale from 0 to 255). Dirt and dirt roads tend to have high intensity, low saturation, and red or orange hue. Vegetation is usually between red and green in hue and has medium to high saturation and varying intensity (depending on time of year). Based on these observations, we are developing rules to segment scenes. We plan to include such variables as time of year, context of current imagery (i.e. on road, off road, etc.), and lighting and weather conditions (sunny, partly cloudy, cloudy, raining, snowing, etc.).

We have applied the rule-based segmentation scheme using the variance over small areas from a corresponding range image and a binary obstacle map as well as the intensity, hue, and saturation from the color image. The portion of the range image overlapping the color image is first "colored" by transforming each point from the range image into the color image plane to determine its RGB values. The resulting image is then rescaled and transformed to the IHS values. A particular pixel from this image may contain several pieces of information from the original range image such as slope, vehicle relative elevation, local elevation, variance, as well as color information of intensity, hue, saturation. With the current implementation, categories such as rough and smooth asphalt, dirt, vegetation, shadows, free-standing height discontinuities, and unknown regions exist.

1.4.4 Texture-Based Segmentation. In addition to the segmentation using intensity, hue and saturation, we have explored the application of a texture-based segmentation procedure developed under Hughes IR&D funds. This procedure was specifically designed for the segmentation of natural terrain images containing highly textured regions such as that shown in the grey level image in Figure 6. Since texture characteristics can only be observed over areas, texture measures need to be calculated over areas or windows within the image. Our approach is to automatically determine a window size and then use this window to calculate a set of texture measure images. A set of thresholds for each measure image is automatically determined by analyzing its corresponding histogram. The thresholded measure images are combined to form a segmentation of the original image. There are artifacts in this image due to the measure values being calculated using windows. Additional processing removes these artifacts to produce the final segmentation.

The optimal window size varies with the correlation distance or period of the texture. A "period slice measure", similar to the one in [6], was developed which uses the placement and direction of thinned edges to estimate period size. The distance between two like oriented edges approximates the texture period at that location. Separate histograms are analyzed to dynamically determine vertical and horizontal sliding window dimensions which can be used for texture measure calculation. The histograms of the individual texture measures are analyzed for threshold selection to produce a set of measure segmentations. These segmentations are combined to produce a preliminary segmentation.

The texture measures chosen correspond to texture characteristics typically found in areas within natural terrain imagery. Measures currently being used include a brightness measure (average intensity over a window), a contrast measure (average edge magnitude over a window), and a coarseness measure (edge count percent over a window). One of the advantages of using such a measure set is that for a given segmented region, not only size, shape, and location information are available, but also a symbolic description of the relative intensity, contrast and density of the region. Since measures chosen correspond to noticeable characteristics within the image, these descriptions will be useful in region identification for image understanding.

One artifact in the process described above is the creation of region bands. These bands, separate regions created at region boundaries, occur when a region boundary area exhibits different characteristics from the regions actually forming that boundary. A technique was designed which eliminates region bands and small regions whose dimensions are on the order of the texture period size.

The texture segmentation algorithm has been applied to color imagery obtained from the ALV data set. The upper lefthand corner of Figure 6 is a grey scale intensity image in which we have averaged the red, green and blue components. This intensity image is used to calculate the average intensity measure image. In order to produce a color edge image the edge magnitude is maximized over the three color planes. The desired segmentation should separate the curved road area from its surroundings, and within the off-road areas, different terrain types should be separated. A 15x31 sliding window was automatically selected by the algorithm for calculation of the average intensity, average edge magnitude and edge density measures. The resulting measure segmentations are shown in the upper right, the lower right, and lower left subwindows of Figure 6, respectively. There are four levels of average intensity in the resulting intensity measure segmentation while the edge density and contrast measure segmentations show only two types of areas in each. The road area is well separated using only the intensity and edge density measures, while the contrast measure helps in differentiating the off-road terrain types. For example, the dark textured area to the right of the road is a low contrast area while the dark area at the top of the image (trees) is a high contrast area. The segmentation produced by combining the three measure segmentations is shown in the left subwindow of Figure 7. The final result obtained after removing the bands is displayed in the right subwindow of Figure 7. The road as well as various terrain types are well separated in the final segmentation.

Preliminary results indicate that a good first pass segmentation can be produced using the algorithm described above. With no a priori knowledge, sliding window dimensions are automatically generated. Then a set of texture measures is calculated for every pixel. Symbolic descriptions corresponding to noticeable texture characteristics are attributed to the resulting regions. Future work will include the development of a set of criteria for segmentation evaluation and a strategy for further processing when appropriate.

1.5 Local Map Construction

The local map is defined as a down-looking, map view of the sensed area local to the vehicle. We have developed two types of local maps: the symbolic local map, and the Cartesian elevation map. The symbolic local map contains important information useful for navigation in symbolic form. Objects such as free standing obstacles, sloped terrain, road edges, ravines, etc. are included symbolically in the local map. For example, a free standing obstacle may have several parameters associated with it such as height, distance from the vehicle, size, shadowed area, color, and location. Each of these attributes are determined from various processing steps or virtual sensors and are accumulated in the local map.

We have also developed a version of the symbolic local map which we call the confidence grid. In this representation, objects found in several scenes using the range and color sensors are placed at the proper location in the grid, incrementing a count at each point when an obstacle exists there. In this way, higher confidence is associated with points having larger counts since a particular obstacle was found at those points over several scenes. We also use several representations for obstacles in the confidence grid including obstacle extents and boundaries (transformed from the spherical coordinates of the range image into the Cartesian coordinates of the map view), tangent points, minimum and maximum distances to the obstacle, and angular extents of the obstacle in the map view. These representations are suitable for map-based planners and vehicle behaviors being developed at Hughes. A symbolic local map of approximately 25 meters by 25 meters is shown in Figure 8. The road is represented in white and the obstacles are outlined in a stripe pattern. The background tiles represent terrain slope with light grey being closest to a horizontal slope and dark grey approaching a vertical slope. Unknown terrain is represented as black. There is unknown area between some of the slope tiles which is the result of a gap between footprints in the laser range scan.

An alternate form of the local map is called the Cartesian elevation map. In this realm, the Cartesian value of a particular point in the range image is calculated, and then the height (Z) at the proper down range (Y) and cross range (X) location is set. The resulting map contains a sparse array of elevation points. To fill in the height at unscanned areas, we interpolate between scanned points and propagate the interpolated values over the elevation map until no holes remain. The Cartesian elevation map is an alternate form for representing the range data and is valuable for implementing obstacle detection methods which would not be possible in the spherical range image coordinate system. Data from multiple range images can also be fused in the Cartesian elevation map so that one map contains information from several different scans. This work is discussed in greater detail in the accompanying range processing paper by Daily et al.

2. Knowledge-Based Target Cueing

Hughes has many years experience in automatic target detection and recognition systems. Many of these traditional approaches are presented in an invited paper titled "Image Understanding Technology and Its Transition to Military Applications" by D.Y. Tseng and J.F. Bogdanowicz. More recently, the AI Center has developed a context cueing system which incorporates scene knowledge in an image interpretation system thereby providing a more reliable classification of objects. Scene knowledge consists of models of objects expected in the scene and their relationships as well as information related to image acquisition. Extracted image features represented as symbolic descriptions and a network of scene object models described using frames drive the interpretation in both a bottom-up and top-down fashion. The interpretation gathers information which provides evidence for or against hypotheses. The system has been exercised on a number of forward looking infrared images and has exhibited performance which exceeds that of traditional approaches. The context currently being exploited includes target motion, spatial relationships between targets (formation knowledge), and spatial relationships between target and background (target/road knowledge). This system is featured at the IU Workshop in an invited paper, "Image Interpretation Using Scene Context", by T.M. Silberberg.

3. Multiprocessor Architecture Applications

To support both reasoning and feature extraction at real time speeds, we require specialized hardware. The computational burden of low level image processing dictates that architectures perform numeric operations at extremely high rates. At the same time, the complexity of symbolic manipulation calls for a general purpose computer architecture. To allow the transformation of numeric data into symbolic objects, these two types of architectures must be efficiently interfaced. Fortunately, both numeric and symbolic processing can be enhanced through parallel computing. The objectives of this IR&D task are to investigate the issues related to data partitioning, task scheduling, and user development environments. The Hierarchical Bus Architecture (HBA) is a specialized multiprocessor machine which Hughes designed and built under IR&D funds in support of this type of vision research. The HBA was presented at the DARPA Image Understanding Architecture Workshop in November, 1986 [7].

Acknowledgements. The progress presented in this overview represents the combined efforts of the technical staff members of the Computer Vision Section. In addition, we would like to acknowledge the useful comments and recommendations received from R. Nevatia and the many valuable ideas that have evolved from discussions with the Autonomous Planning Systems staff at the AI Center, in particular D.W. Payton and D.M. Keirsey. We thank D.Y. Tseng, Manager of the AI Center, for his continuous support of our technical pursuits.

References

[1] D.W. Payton, "An Architecture for Reflexive Autonomous Vehicle Control", *IEEE Conference on Robotics and Automation*, San Francisco, CA, April, 1986, Vol. 3, pp. 1838-1845.

[2] R.S. Wallace, A. Stentz, C. Thorpe, H. Moravec, W. Whittaker, T. Kanade, "First Results in Robot Road Following", *Proceedings of the Nineth International Joint Conference on Artifical Intelligence*, Los Angeles, CA, August, 1985, pp. 1089-1095.

[3] L.S. Davis, T.R. Kushner, J.J. LeMoigne, A.M. Waxman, "Road Boundary Detection for Autonomous Vehicle Navigation", *Optical Engineering*, Vol. 25 No. 3, 1986, pp. 409-414.

[4] Martin Marietta Denver Aerospace, "The Autonomous Land Vehicle Second Quarterly Report", September, 1986, pp. 11-19.

[5] Y. Ohta, T. Kanade, and T. Sakai, "Color Information for Region Segmentation", *Computer Graphics and Image Processing*, Vol. 13, 1980, pp. 222-241.

[6] F.M. Vilnrotter, R. Nevatia, and K.E. Price, "Structural Analysis of Natural Textures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8, 1986, pp 76-89.

[7] R.S. Wallace, "Swath Algorithms for Vision", *DARPA Image Understanding Architecture Workshop*, McLean, VA, November, 1986.

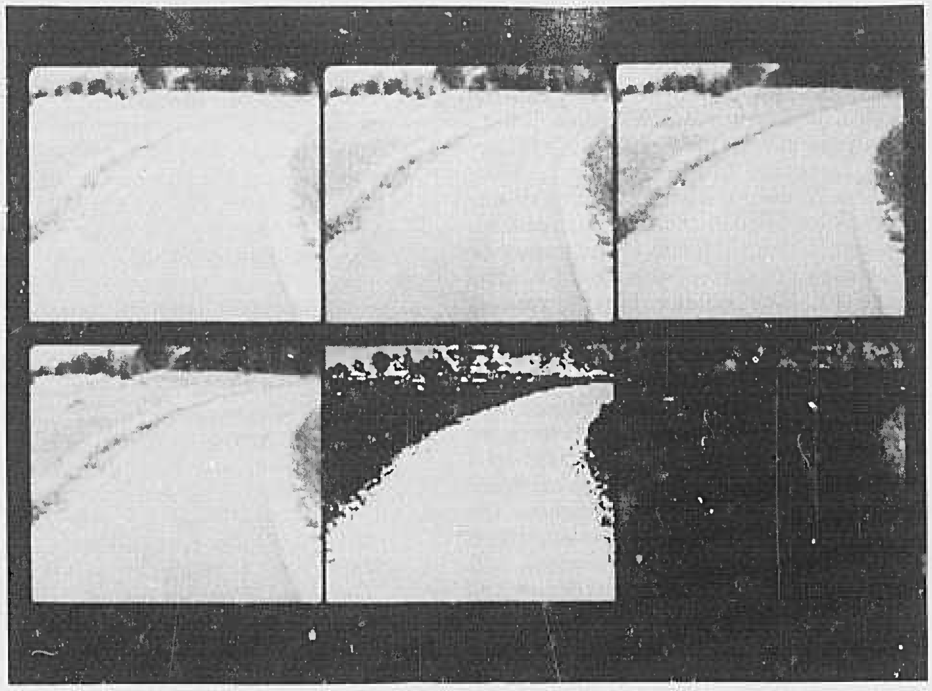


Figure 3
Top row: Red, green, and blue planes.
Bottom row: Intensity, hue, and saturation planes.

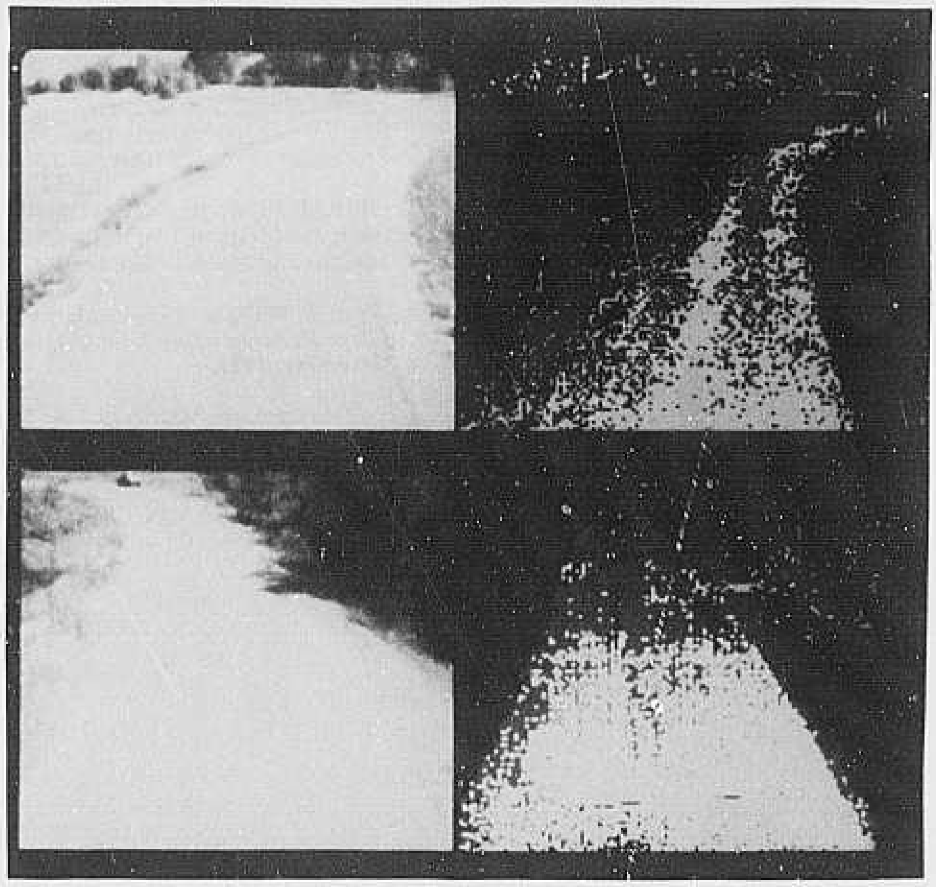


Figure 4
Bayes classification of two scenes into four classes: sunny road (white), cloudy road (light grey), dirt (dark grey), and vegetation (black).

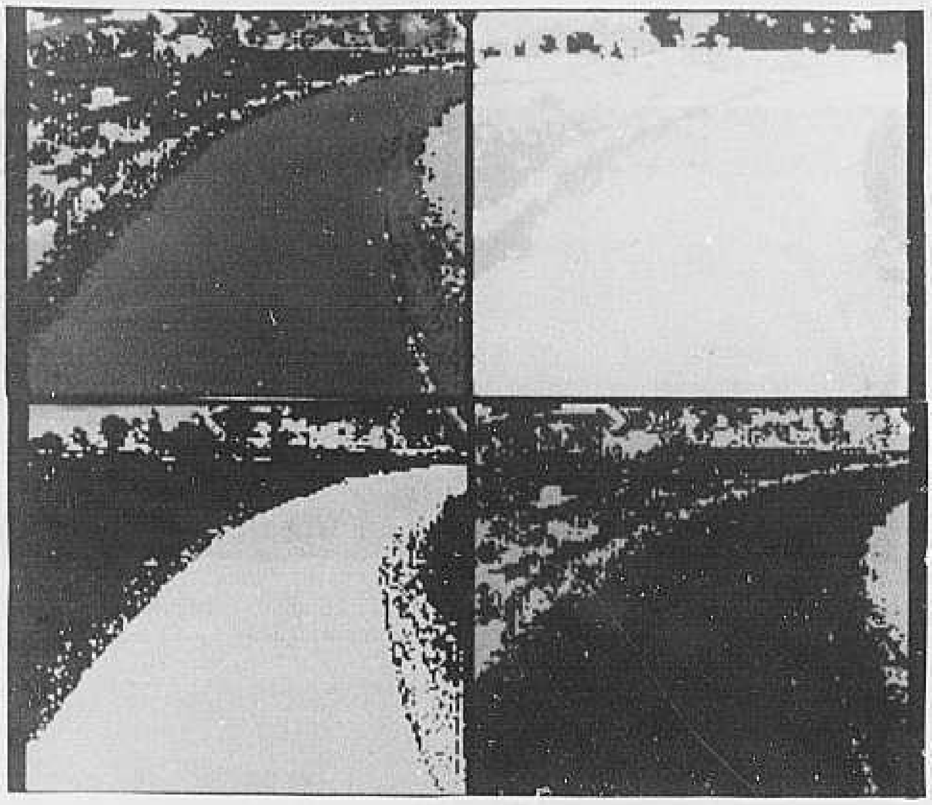


Figure 5

Top (left;right): Composite quantifieation in IHS space; 3 levels of intensity segmentation
Bottom (left;right): 12 levels of hue segmentation; 3 levels of saturation segmentation

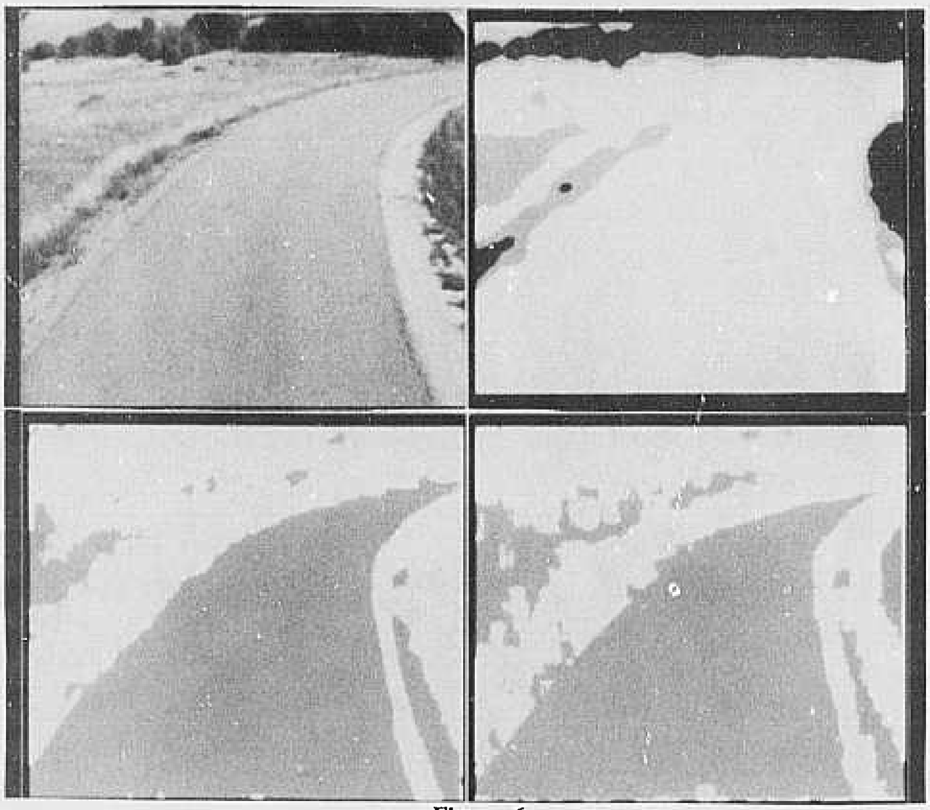


Figure 6

Top (left;right): Image from averaged RGB; Average intensity (brightness) segmentation
Bottom (left;right): Edge density (coarseness) segmentation; Average edge magnitude (contrast) segmentation

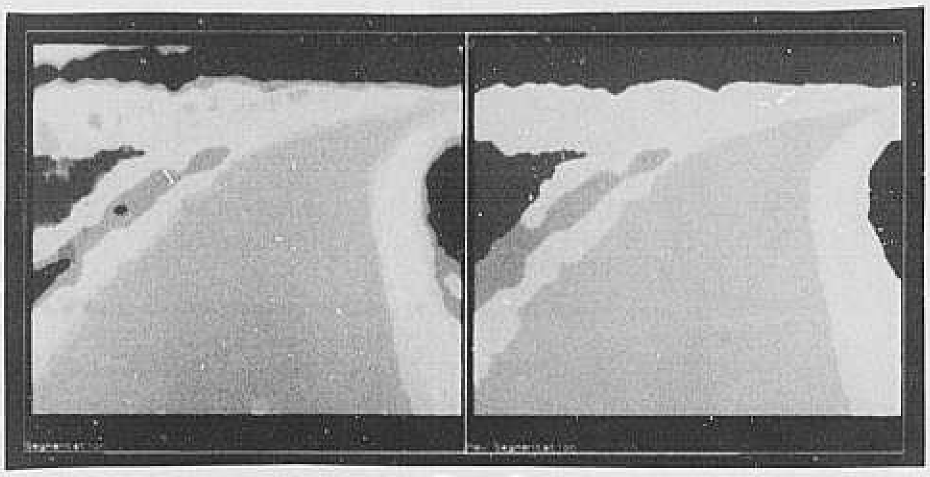


Figure 7

Left: Preliminary segmentation combining brightness, coarseness, and contrast segmentations.

Right: Final segmentation

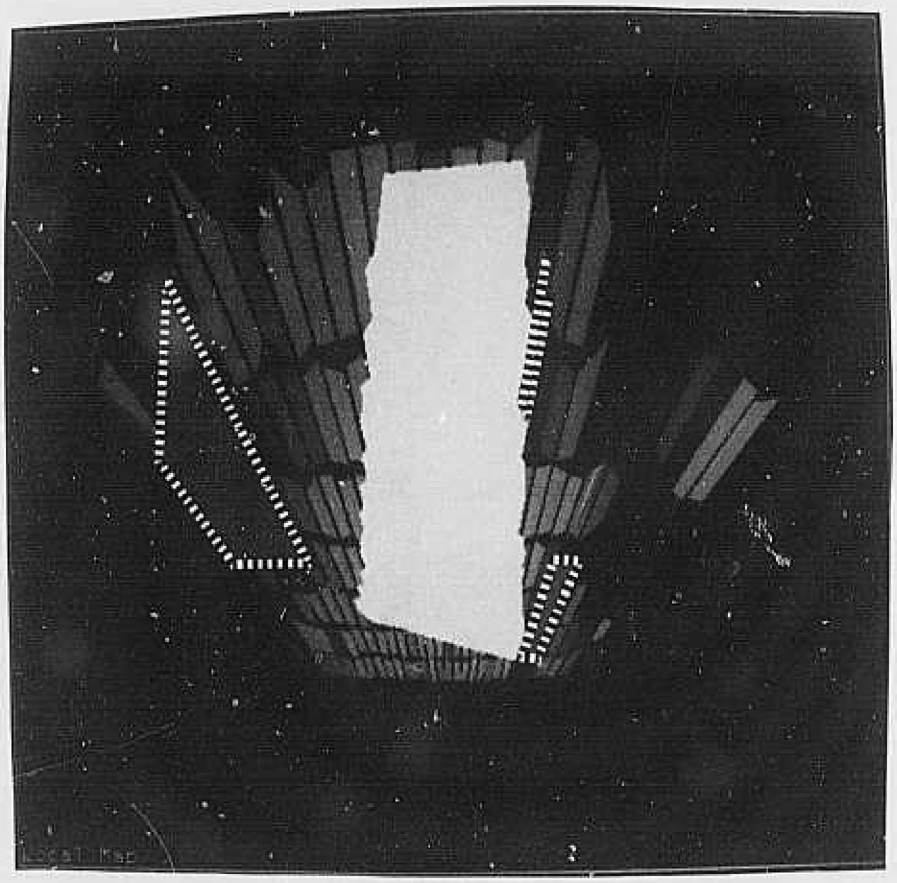


Figure 8.
A symbolic local map displaying the road, obstacles, and terrain slope information.

DETECTING OBSTACLES IN RANGE IMAGERY

by Michael J. Daily, John G. Harris, and Kurt Reiser

Hughes Artificial Intelligence Center
23901 Calabasas Road, Calabasas, CA 91302

Abstract

Outdoor natural environments pose significant problems to the task of vehicle navigation, among the most difficult being obstacle detection and avoidance. Detecting and avoiding obstacles requires three dimensional sensing of the environment through stereo or "shape from" techniques or laser range devices. We present both a theoretical, ideal range sensing paradigm and an actual laser range sensor and their respective capabilities and limitations. We discuss the definition of obstacles in natural environments and develop a method for locating such obstacles which uses a vehicle model to determine local regions of traversability. Several alternative methods for extracting important features of obstacles are discussed in the context of the obstacle definition, the ideal sensor imagery, and the range imagery from the existing sensor. We also present results for two scenes of difficult outdoor terrain containing features such as ravines, steep slopes, trees, rocks, deep grass, and man-made objects.

1. Introduction

Our goal is to build the perception capability for an autonomous vehicle which can navigate successfully around obstacles using range sensors. We define obstacles as any nontraversable location in space. Our approach first studies the limits of obstacle detection in ideal range imagery. We then show the effects of corrupting the ideal range data with nonideal constraints. In particular, we study the use of the Environmental Research Institute of Michigan (ERIM) laser range scanner applied to complex, outdoor scenes with features such as trees, ravines, and sloping terrains. The novel idea in our approach is that we purposely steer away from symbolic descriptions of our scene. We find obstacles at the signal level and apply our formal definition of obstacles in that domain. We also describe other more conventional, but faster, approaches to approximately find steep sloping regions or large discontinuities in range or height.

Range data can be obtained from techniques using stereo, structured light, triangulation, structure from motion, time of flight, and range from focusing. Most of the work to date in range image processing focusses on indoor applications with simple polyhedral objects, which limits its usefulness for natural outdoor scenes. Duda et al. [4] discuss the use of range and corresponding reflectance images to find planar surfaces, using assumptions of horizontal and vertical man-made surfaces to direct their algorithm. Milgram and Bjorkland [11] transform the range data from spherical coordinates into Cartesian coordinates, fit planes to 5 x 5 windows, and form planar regions using a variety of constraints including surface normals and fitting error. They also propose incorporating the technique into a vehicle navigation system. Henderson [7,8] develops a method for extracting planar faces from range data which groups 3-D points into a local spatial proximity graph and derives polyhedral surfaces. His method works on either dense or sparse data and applies to both range image segmentation and object recognition. Besl and Jain [3] propose using Gaussian and mean curvature to invariantly characterize intrinsic and extrinsic properties of smooth surfaces for the purpose of object recognition and surface description. For an excellent survey of additional work see Besl and Jain [2].

Several other attempts at range image processing more directly related to outdoor, natural scenes and autonomous vehicle navigation have been made. Lewis and Johnston [9] of JPL assume obstacles occur only at range discontinuities. As the authors point out, this method fails to find obstacles with smooth rounded edges. More recently, Zuk et al. [15] attempt to locate roads using range and reflectance data from their laser range device, and apply this knowledge to autonomous navigation. Their methods use texture to differentiate between road and off-road regions. Hebert and Kanade [6] segment ERIM images of park scenes based upon local geometric surface properties. Their algorithm labels regions of a Cartesian map as either *smooth* or *obstacle* based upon surface curvatures. Hughes [14] develops several techniques for correcting ERIM range data, locating obstacles on flat ground, calculating approximate slope over small patches in three-space, and fusing range data with corresponding color video imagery. Martin Marietta Corporation (MMC) [10] also develops obstacle detection algorithms for use with the Autonomous Land Vehicle (ALV). They subtract the current range scene from pre-stored range images of planar ground at various scanning angles to produce a binary obstacle map. Our work primarily differs from these attempts in that we formally define obstacles.

2. The Ideal Range Sensor

For our purposes, the three dimensional world is made up of piecewise continuous surfaces described by some $W(x,y,z) = 0$. An autonomous vehicle must represent its environment in terms of a 3-D spatial occupancy map, marking points in three-space where it may safely travel. We assume that our vehicle is a land vehicle, and therefore can travel only along the surface of the world. For illustrative purposes, assume the 3-D world is a sphere resting on planar ground.

We would like a range sensor to return the exact distance to every point in its field of view. The ideal sensor would provide a continuous description of the world with infinite accuracy and give us a theoretical framework in which to study obstacle detection in range imagery. There are two major reasons for discussing the ideal sensor and its implications: to point out the limitations of detecting obstacles in range imagery, and to generalize obstacle detection to use any source of range imagery. The ideal sensor is a continuous function, $\rho(\theta, \phi)$, which gives the exact range to the point nearest the sensor in directions specified by θ and ϕ. The geometry of this sensor is shown in Figure 1. Since the sensor is assumed to be at the focal point in the imaging geometry, the perspective effect produces occluded or unscanned areas. These occluded areas are the only limitation of our ideal data.† Occluded regions, marked by sudden discontinuities in range, occur where the directional vector specified by θ and ϕ lies in the plane tangent to the sensed surface (for smooth surfaces only). That is, there is an occluded surface when either of the following is true:

$$\frac{\partial \rho}{\partial \theta} \rightarrow \infty$$
$$\frac{\partial \rho}{\partial \phi} \rightarrow \infty$$

These discontinuity conditions define continuous occluding contours in $\rho(\theta, \phi)$ space. As will be discussed shortly, occluded areas are important in developing alternate forms for representing the data and directing processing of the range imagery. Figure 2 shows the ideal range image of the sphere resting on planar ground. Figure 3 shows the corresponding occluding contour.

In order to directly apply our obstacle definition to the ideal range image, we must first reconstruct the three dimensional Cartesian world from the ideal range image. We can obtain parametric Cartesian descriptions of the surfaces using the following coordinate transformations:

$$x(\theta, \phi) = \rho(\theta, \phi) sin\theta$$
$$y(\theta, \phi) = \rho(\theta, \phi) cos\theta cos\phi$$
$$z(\theta, \phi) = \rho(\theta, \phi) cos\theta sin\phi$$

† Given vehicles with nonzero stopping times and finite turning speeds, this limitation sets the maximum speed a vehicle may travel over a given terrain.

This parametric description of the surface is identical to the original $W(x,y,z)$ for all visible surfaces.

As mentioned earlier, we define an obstacle as any location in three-space where a particular vehicle can not be placed and maintain a stable configuration. A stable configuration includes at least all of the following in the ideal case:

- sufficient clearance of the vehicle undercarriage
- wheel locations in 3 space within the tolerance of the vehicle suspension
- vehicle base at a slope less than the maximum allowable slope
- vehicle height not exceeding the height of overhanging objects

Other variable factors which influence each of the above constraints include vehicle heading or orientation, speed, risk factors, mission goals, and weather conditions. Additional considerations include vehicle suspension, weight distribution, undercarriage clearance, and size. A perfect obstacle detection procedure must apply a complete vehicle model to a three dimensional representation of the world produced using the theoretical, ideal range sensor. The three dimensional world may be represented in at least two equivalent forms to accomplish this task: the range image plane, or the Cartesian 3-D world. In either case, as was mentioned earlier, the only limitation of the ideal sensor is unknown (unscanned) areas which are occluded from the sensor.† By moving the completely modeled vehicle through the three dimensional world volume at every possible heading and speed (and using all variable factors), we can produce a traversibility map of the sensed world with complete confidence. We call this the perfect or ideal obstacle detection technique.

3. A Real Range Sensor

As was seen, the only real limitation of the ideal sensor as defined was the sensor's inability to scan the unknown regions. Realistically, there are many other limitations to range sensors. We now discuss the problems associated with data obtained from a real sensor, the ERIM laser range scanner. ERIM range imagery contains significant degradations from our idealized case; examining these degradations will facilitate an understanding of both the advantages and inherent limitations of our techniques.

† The vehicle model can not be applied with certainty at any location where unknown terrain exists, although we can set bounds on the unknown area allowing tentative, low confidence decisions to be made at such points.

- Range Resolution: In the ideal case the exact range (ρ) in any specified direction was known to infinite precision. The ERIM scanner records range as an eight bit value which places a lower bound on the size of features we can detect. Additionally, discretization error introduces wavelike patterns in flat surfaces.
- Maximum Range: The ERIM scanner measures distance modulo 64 feet. For example, a range measurement of 8 represents only one of several possible distances of the form $8+64n$ feet, where $n = 0, 1, 2...$ This causes ambiguity in the range data.
- Beam Divergence: In our idealized imagery the range to any visible point was available. In contrast, since the ERIM scanner uses a divergent laser ray to measure range, the images contain distances to visible *areas*. A divergent beam illuminates an ellipsoidal patch called a "footprint." The farther a laser ray travels, the larger its footprint size. Consequently, distant objects are much less finely resolved.
- Angular Discretization: The ERIM sensor measures distance along predetermined directional vectors. Most of our imagery, from the ERIM scanner on board the Autonomous Land Vehicle at Martin Marietta Corporation in Denver, employs 64 different values of ϕ and 256 different values of θ to generate a 64 x 256 range image. Constant increments between values of θ and ϕ are used. Consequently, a map view elevation image generated from this data using a point model for the laser footprint will be sparsely populated in areas distant from the scanner. Angular discretization and beam divergence reduce our feature resolution as a function of distance and potentially remove higher frequency events.
- Reflective Surfaces: The ERIM sensor actually determines distance by comparing the phase shift in source and reflected signals. When a laser ray strikes a reflective surface, a smooth surface at a glancing angle, or nothing at all, little or no reflected signal is available. Erroneous or random range values are recorded at these locations.
- Mechanical Positioning: Laser rays are deflected through various angles in the ERIM device via rotating mirrors. Jitter and positioning errors in these mirrors cause distortion of scene features. Larger problems are introduced by motion of the vehicle during the scanning process.

Figures 4a and 4b show two range images of terrain at Martin Marietta in Denver. They have been corrected (as described in [14]) so that the ambiguity levels mentioned above are no longer present. Figure 4a is of a winding dirt road sloping down between two large trees on either side of the road. The right foreground contains grass and small rocks while the left foreground drops off into a ravine, the opposite side appearing at the left top corner of the image. Figure 4b is of an asphalt road gently sloping down between two gates. On the right side of the road is a ditch, with the terrain sloping up on the other side of the ditch, while the left side slopes down. The top of the image contains "sky" where the scanner returned a random value since nothing was present.

4. Detecting Obstacles with a Real Range Sensor

In Section 2, we described the ideal obstacle detection technique which applies a complete vehicle model to the continuous data from the ideal sensor. However, the complexity of such an obstacle detection scheme warrants consideration of successively simpler approximations which are feasible using the real sensor. By relaxing the constraints and variables assumed for the ideal case, we can apply more realistic techniques for obstacle detection to the actual range data.

4.1 The Vehicle Model

We assume that the vehicle model is applied to a Cartesian description of the world and not the range image plane. This is important since modelling the vehicle in the range image plane requires varying its size over the image, a potentially costly approach. In the theoretical case, the three dimensional Cartesian world is merely an alternate, continuous form for representing the continuous range data from the ideal sensor. To simplify processing, we use the assumption that the 3-D world is a single valued function, $z(x, y)$, of the two map directions, where z is height. We call this the Cartesian elevation map. Where multiple heights occur at a particular (x, y) location, the highest point is kept. Ignoring beam divergence and the footprint model of the real sensor, and assuming the range data is sampled using a point model, the elevation map will be sparsely populated. To fill the empty regions resulting from the sparse data, we have developed a simple linear interpolation algorithm which operates only on scanned regions (as opposed to occluded regions) and interpolates between actual scanned points [5]. The resulting elevation map approximates the original world $z(x, y)$. Figure 5 shows the Cartesian elevation map for the ideal range image of Figure 2. The occluding curve from Figure 3 is mapped to the unknown area in the elevation map, shown in black. Note that the sphere in the 3–D world actually represents a portion of a hemisphere on top of a cylinder in the elevation map of Figure 5.

Assuming the world is defined by the single valued function $z(x, y)$ removes the fourth constraint for a stable configuration, that of vehicle height (e.g. space beneath overhanging objects is removed). Allowing only heading or orientation to vary and ignoring vehicle weight, we arrive at our first approximation to the best technique: a vehicle model with suspension, clearance, and slope constraints. Suspension is modeled as N springs (where N is the number of wheels) with only one degree of freedom, compression, and attached to a rigid planar body representing the vehicle undercarriage (see Figure 6). Wheels are assumed to be points at the end of each spring. Applying this vehicle model to the elevation map will produce points of intraversibilty where the ground exceeds vehicle clearance, maximum slope, or suspension at multiple headings or orientations. The result may be thought of as a three dimensional traversability map indicating obstacles at different headings for each (x, y) location.

The actual implementation we chose is simple, and approximates the potentially complex force equations required for the spring suspension model of rigid body vehicles with several wheels. We fit a plane to the ground points touching the wheels using a standard least squares technique and, since the fitted plane will not pass through all the points in most cases, leaving some wheels below the ground and others above the ground, we move the fitted plane vertically up by the sum of the maximum distance any wheel is above the plane and the minimum distance tolerated by the suspension between a wheel and the undercarriage, maintaining the same plane normal. The fitted plane then corresponds to the undercarriage of the vehicle within some tolerance for the suspension, and no point on the ground where a wheel touches is above the plane. Slope of the undercarriage is available directly from the normal to the plane, and the distance of each wheel beneath the undercarriage corresponds to the suspension tolerance for the vehicle. Ground clearance is checked by simply calculating which side of the planar undercarriage each point beneath the vehicle is on. Points above the undercarriage exceed the clearance. Ground clearance specifications vary for vehicles with different numbers of wheels.† The algorithm checks the slope, clearance, and suspension at each point in the Cartesian elevation map and for each heading desired.

Results for this technique applied to Figures 4a and 4b are shown in Figures 7a-d and 8a-d. Excluding preprocessing and time to build the elevation map, this technique requires $O(n^2 lw\mu)$ inner loop operations where n is the size of the Cartesian elevation map (assumed to be square), l is the length of the vehicle, w is the width, and μ is the number of different orientations. For the example of Figure 7a, the parameters are n=85 (9 inch resolution), l=18 (13.5 feet), w=12 (9 feet), and μ=1 (one orientation pointing from the top to the bottom of the image).

Given the complexity of the above algorithm (1,560,600 operations in the example), we chose to implement a yet simpler approximation to the ideal obstacle detection method. Removing the constraint on vehicle suspension and keeping the constraints on slope and clearance, we apply a two wheeled "bike" to the Cartesian depth map at each location and at several orientations. The two wheeled bike is far simpler computationally since all that is required is two dimensional slope calculation of a line and checking for points above the line which exceed the bike's clearance. However, since the two wheeled bike is applied at a discrete number of orientations, it is possible to miss certain obstacles which a square vehicle would find. To remedy this problem, a simple convolution style post-processing step which checks for obstacle points under a square mask can be used to produce a conservative traversability map for rectangular vehicles. The two wheeled method is over five times as fast as the multiple wheeled technique, with similar results.

† For example, certain tracked vehicles or vehicles with several wheels on each side do not have clearance problems along the length of the vehicle, but do across the width between tracks. In these cases, we do not check clearance along the left and right sides of the vehicle.

4.2 Discontinuities

While the only fail-safe method for locating all intraversable areas uses the complete vehicle model, due to the complexity of such a model, sub-optimal techniques are also reasonable to pursue. An alternate approach to applying the vehicle model to the three dimensional data is to extract specific features which always signal an obstacle. Two useful features for obstacle detection are discontinuities and slope. Detecting these features in $\rho(\theta, \phi)$ range image space is advantageous for two reasons: algorithms are fast since no transformations need to be done, and there is no need to represent unknown regions since all points represent scanned terrain. Earlier, we discussed the characteristics of occluded regions for the ideal range data. These occluded regions also represent significant discontinuities in range. For real range imagery, the discrete formulation is similar to the earlier continuous case for occluded regions, that is when

$$\frac{\Delta\rho}{\rho} > threshold$$

then a range discontinuity exists, where $\Delta\rho$ is calculated in both the θ and ϕ directions (i.e. separately across rows and up columns of the range image). The above ratio, rather than just $\Delta\rho$, is used to normalize the change in range with respect to distance, since $\Delta\rho$ will be small close to the sensor and larger farther away even on flat ground. Depending on the value chosen for *threshold*, we may falsely signal very oblique planes as discontinuities. Bergman and Cowan [1] use the same technique, while Mitiche and Aggarwal [12] propose a more sophisticated jump edge detector in range imagery which analyzes the first order differences of range values. This approach handles the steep plane problem better than our method, but is not used for two reasons. First, oblique planes are not that common in our natural outdoor imagery. Second, when these planes do occur we do not put much credibility in them since they are sparsely sampled along their length. Also, these types of planes are very succeptable to noise since they reflect most of the incident beam away from the sensor. Either method will miss small jump discontinuities, meaning that small occluded regions will be overlooked. Figures 9a and 9b show discontinuities in range (white areas) for Figures 4a and 4b.

We have just seen how discontinuities in range are useful for detecting unknown or unscanned areas; discontinuities in height signal abrupt protruding or depressed obstacles. We would like to calculate the maximum change in height over a specific neighborhood of the range image. In the continuous realm, discontinuities in height are present when the following is true:

$$\sqrt{(\frac{\partial z}{\partial \theta})^2 + (\frac{\partial z}{\partial \phi})^2} \rightarrow \infty$$

where z, the height, is calculated as

$$z = \rho(\theta, \phi) cos\theta sin\phi$$

Note that this formulation for the continuous case of discontinuities in height is equivalent to the previous treatment of discontinuities in range.

However, discontinuities in height in the discrete case provide useful information not available from discontinuities in range. Abrupt changes in height over small, discrete neighborhoods need not signal changes in range, as in the case of scanning up a vertical surface. In such a case, a range discontinuity will occur only if the sensor scans over the top of the object and hits ground behind the object. Our implementation for the real range data calculates the difference between the maximum and minimum height over a small constant size neighborhood. If the maximum height difference at each point exceeds a height threshold for obstacles (determined with respect to the vehicle), then that point is marked as obstacle and is intraversable. Since the real range imagery has limited resolution which becomes worse as the distance increases from the scanner, a particular constant size neighborhood in the range image close to the scanner covers a much smaller region in three-space than the same neighborhood farther away. Normalizing the change in height with respect to distance or varying the neighborhood size are both options which circumvent this problem; however, the main advantage to using constant size neighborhoods is ease of implementation and speed. Figures 10a and 10b show results for detecting discontinuities in height for the same two range images.

4.3 Slope

Another approximation to the ideal obstacle detection technique uses slope, since slope of the scanned terrain is a useful feature for determining traversability. Viewed as a global measure, slope provides a general description of the world of the vehicle. Applied locally, slope describes smaller, higher frequency characteristics of the environment. A natural method for representing these slope features is a multi-resolution slope map, with several layers of decreasing resolution beginning with fine, high frequency slopes and progressing to global, coarse slopes. Our implementation of the multi-resolution slope map consists of four levels of resolution beginning with the full range image (256 x 64) and ending with the coarsest map of size 32 by 8. At the coarsest level processing speed and smoothing of noise are best, while successively lower levels provide more accuracy at the cost of speed and increased noise. Coarser level range images are obtained by averaging the range values over small neighborhoods and sub-sampling by factors of two in each direction. In practice, we have used several methods for calculating the slope or the surface normal of a patch of scanned ground, each with its own advantages and disadvantages, including least squares in angle, angle range image space [6], least squares in Cartesian space, and others [14]. A particularly direct method for calculating slope simply computes the surface normal using the cross product of two vectors defined by four neighbors in the range image. Results for the multi-resolution technique using the cross product are shown in Figure 11. Brighter areas in the bottom set correspond to steeper slopes.

A fast slope approximation technique is to find the slope of the image in the radial direction. Figure 12 shows the geometry for calculating the slope between successive tilt angles. For each value of θ we find:

$$tan\psi = \frac{\rho(\phi + \Delta\phi) sin\Delta\phi}{\rho(\phi + \Delta\phi) cos\Delta\phi - \rho(\phi)}$$

Once ψ is known, we can calculate β, the angle the tangent line makes with the ground, by noting that $\beta = \psi - \phi$. For the case of a piecewise continuous $\rho(\theta, \phi)$, we take the limit of the above equation:

$$\lim_{\Delta\phi \rightarrow 0} tan\psi = \frac{\rho}{d\rho / d\phi}$$

Previously, we stated that the criterion for detecting discontinuities in range along ϕ is $\partial\rho / \partial\phi \rightarrow \infty$. The limiting case above is the more general form of this criterion. Figure 13 shows results for slope from the cross product and in the radial direction for both scenes. White points are above 15 degrees.

5. Conclusion

We formally defined obstacles as intraversable areas for a given vehicle, thus avoiding *ad hoc*, incomplete definitions. We discussed the limitations of the theoretical range sensor, and presented the ideal obstacle detection method which uses a complete vehicle model applied to the continuous range data. The problems associated with a real laser range scanner, several approximations to the perfect obstacle detection technique, and results of each technique applied to difficult, natural terrain were presented.

The formal definition of an obstacle serves two purposes. Firstly, the definition leads to a new obstacle detection algorithm which we have described and implemented. Secondly, the formal definition allows us to compare and evaluate how well our faster, but approximate, obstacle detection algorithms work. We have sought to generalize our discussion of obstacle detection. Consequently, the ideal sensor analysis and obstacle detection ideas developed in this paper apply to all types of range data regardless of the sensor or method used to obtain the data. However, the practical algorithms we devise rely on features of laser range scanners, specifically the ERIM range scanner. While the vehicle model provides the most complete description of obstacles in the environment, discontinuities and slope are adequate in many situations.

Acknowledgements

This work was funded by the Defense Advanced Research Projects Agency (DARPA) under the Knowledge Based Vision Techniques contract (DACA76-85-C-0007).

We thank Martin Marietta and ERIM for making the range data available. Ram Nevatia provided many helpful suggestions. The Hughes AI Center staff and, in particular, Tom Caudell, Karen Olin, Teresa Silberberg, Rich Wallace, and Felicia Vilnrotter were helpful in producing this paper.

Bibiliography

[1] A. Bergman and C.K. Cowan, "Noise-tolerant range analysis for autonomous navigation," in *Proc. 5th Nat. Conf. on Artificial Intelligence*, vol. 2 Philadelphia, PA, Aug. 11-15, 1986, pp. 1122–1126.

[2] P.J. Besl and R.C. Jain, "Invariant surface characteristics for 3-D object recognition in range images," *Computer Vision, Graphics, and Image Processing*, vol. 30, pp. 30–80, Jan. 1986.

[3] P.J. Besl and R. C. Jain, "Three-dimensional object recognition," *ACM Computing Surveys*, Vol 17, No 1, pp 75–145, March 1985.

[4] R.O. Duda, D. Nitzan, and P. Barrett, 'Use of range and reflectance data to find planar surface regions," *IEEE Trans. Pattern Anal. Machine Intell.* PAMI-1, pp. 254–271, July 1979.

[5] J.G. Harris, "The coupled depth/slope approach to surface reconstruction," Tech. Report 908, MIT Artificial Intelligence Lab, 1986.

[6] M. Hebert and T. Kanade, "First results on outdoor scene analysis using range data, " *Proc. Image Understanding Workshop*, pp. 224–231, Dec. 1985.

[7] T.C. Henderson, "Efficient segmentation method for range data," in *Proc. Soc. Photo-Optical Eng. Conf. on Robot Vision*, vol. 336, Arlington, Va., May, 1982, SPIE, Bellingham, WA., pp. 46–47.

[8] T.C. Henderson, "Efficient 3-D object representations for industrial vision systems, " *IEEE Trans. Pattern Anal. Machine Intell.* PAMI-5,6, pp. 609–617, Nov. 1983.

[9] R. A. Lewis and A. R. Johnston, "A scanning laser rangefinder for a robotic vehicle," *Proc. 5th Int. Joint Conf. Artificial Intelligence*, vol. 2, Cambridge, MA., Aug. 1977, pp. 762–768.

[10] Martin Marietta Corporation, "The autonomous land vehicle second quarterly report," MCR–84–600, Sept. 1986.

[11] D.L. Milgram, and C.M. Bjorkland, "Range image processing: planar surface extraction," in *Proc. 5th Int. Conf. Pattern Recognition*, Miami Beach, FL, Dec. 1980, pp.912–919.

[12] A. Mitiche, and J.K. Aggarwal, "Detection of edges using range information," *IEEE Trans. on Pattern Analysis and Machine Intellignece*, Vol. 5, No. 2, pp.174–178, Mar. 1983.

[13] D. Nitzan, R. Bolles, J. Kremers and P. Mulgaonkar, "3D vision for robot applications," NATO Workshop on Knowledge Engineering for Robotic Applications, Maratea, Italy, May 12–16, 1986.

[14] D.Y. Tseng, M.J. Daily, K.E. Olin, K. Reiser, and F.M. Vilnrotter, "Knowledge-based vision techniques annual technical report, " ETL–0431, U.S. Army ETL, Fort Belvoir, VA., March 1986.

[15] D. Zuk, F. Pont, R. Franklin, and M. Dell'Eva, "A system for autonomous land navigation," Environmental Research Institute of Michigan, IR–85–540, Nov. 1985.

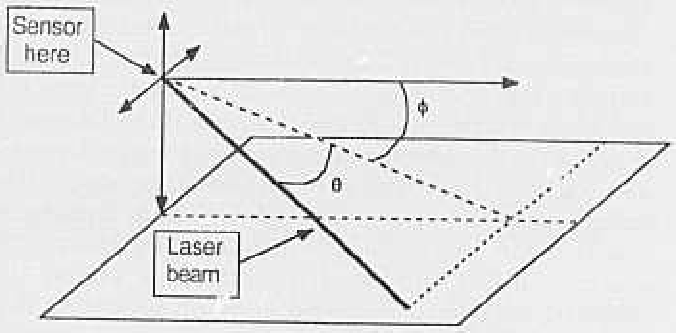


Figure 1. Range sensor geometry.

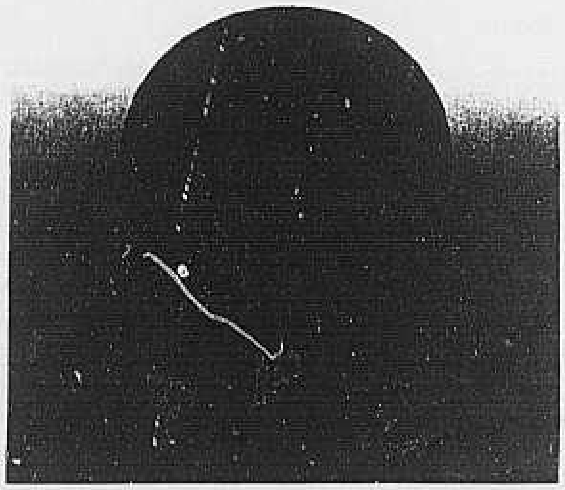


Figure 2. Range image of sphere. Brightness corresponds to distance.

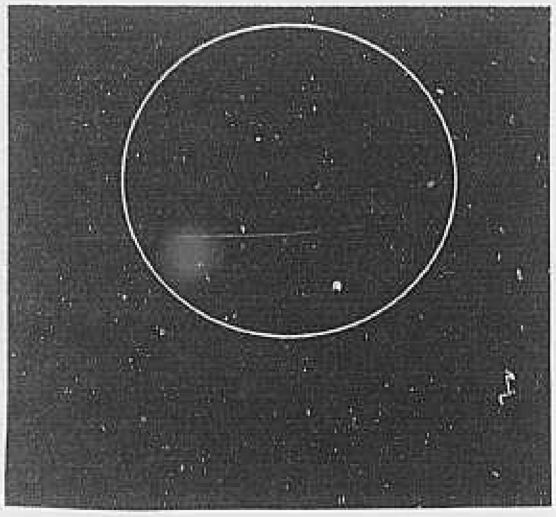


Figure 3. Occluding contour of sphere.

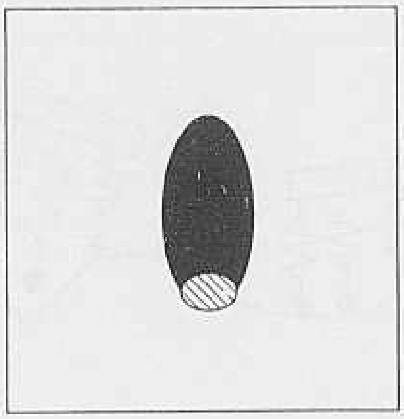


Figure 5. Cartesian elevation map of sphere. Black area is unknown, shaded area is visible portion of sphere, white is visible portion of plane.

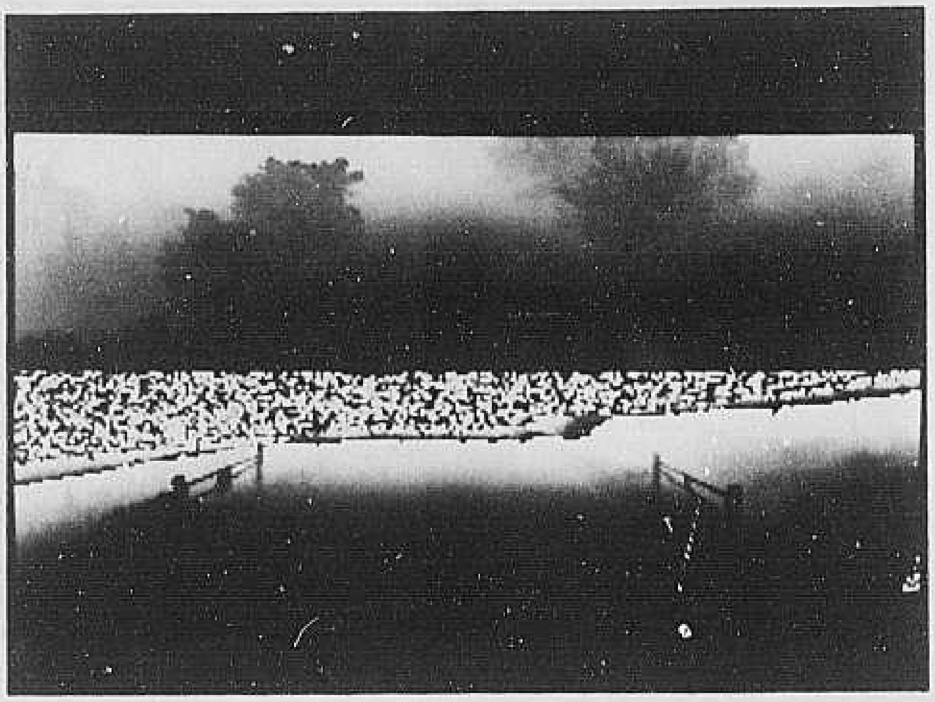


Figure 4. (a) Range image of tree scene, (b) Range image of fences.

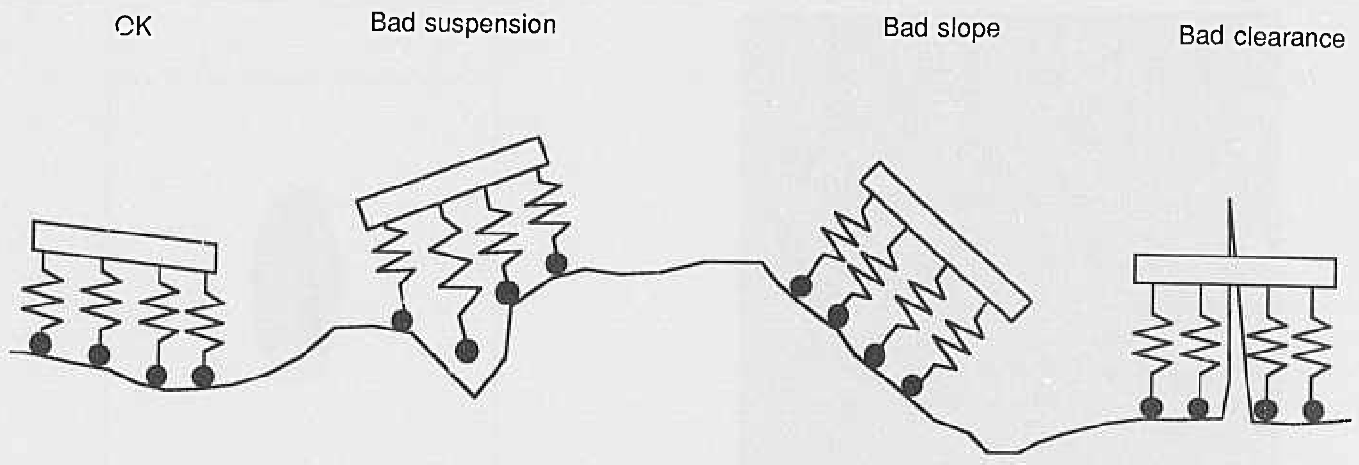


Figure 6. Spring model of vehicle.

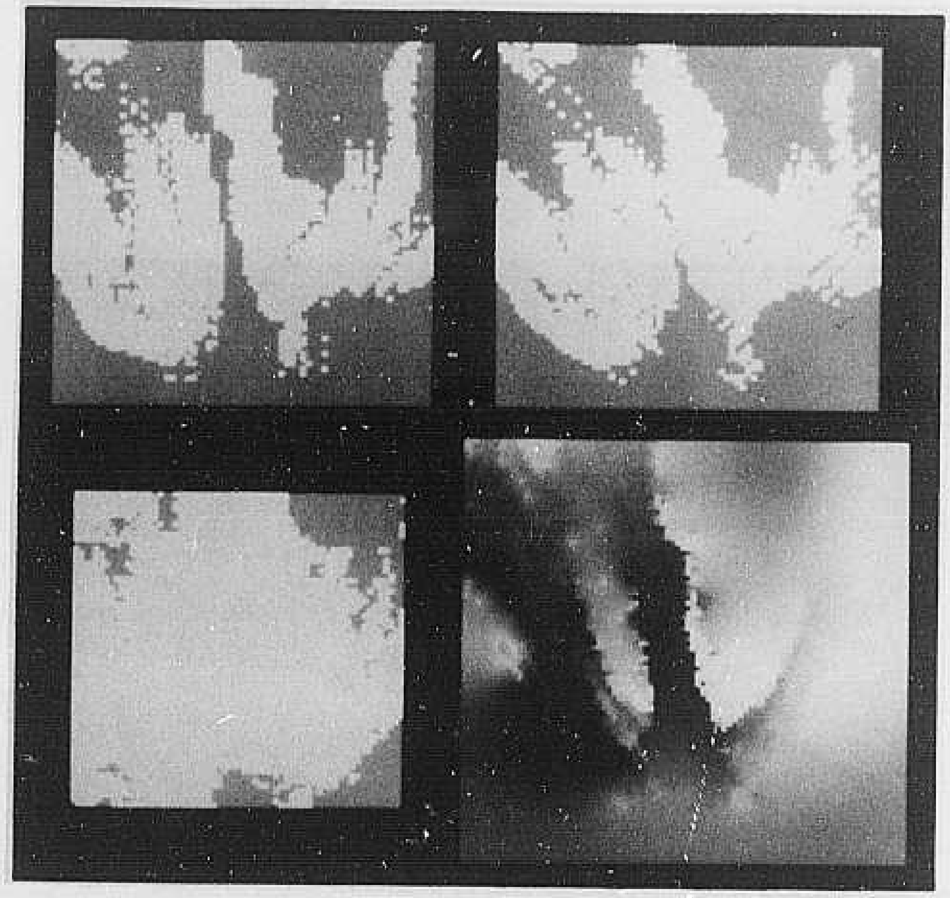


Figure 7. Traversability analysis of tree scene using vehicle model. (a) Bottom-right; Cartesian elevation map of terrain (scanner at bottom). Brightness corresponds to height. (b) Bottom-left; large 8 wheeled vehicle (13.5 ft. by 9 ft.) at one orientation. White is intraverable. (c) Top-left; small 8 wheeled vehicle (7 ft. by 5 ft.) at one orientation. (d) Top-right; small 8 wheeled vehicle at four orientations (every 45 degrees).

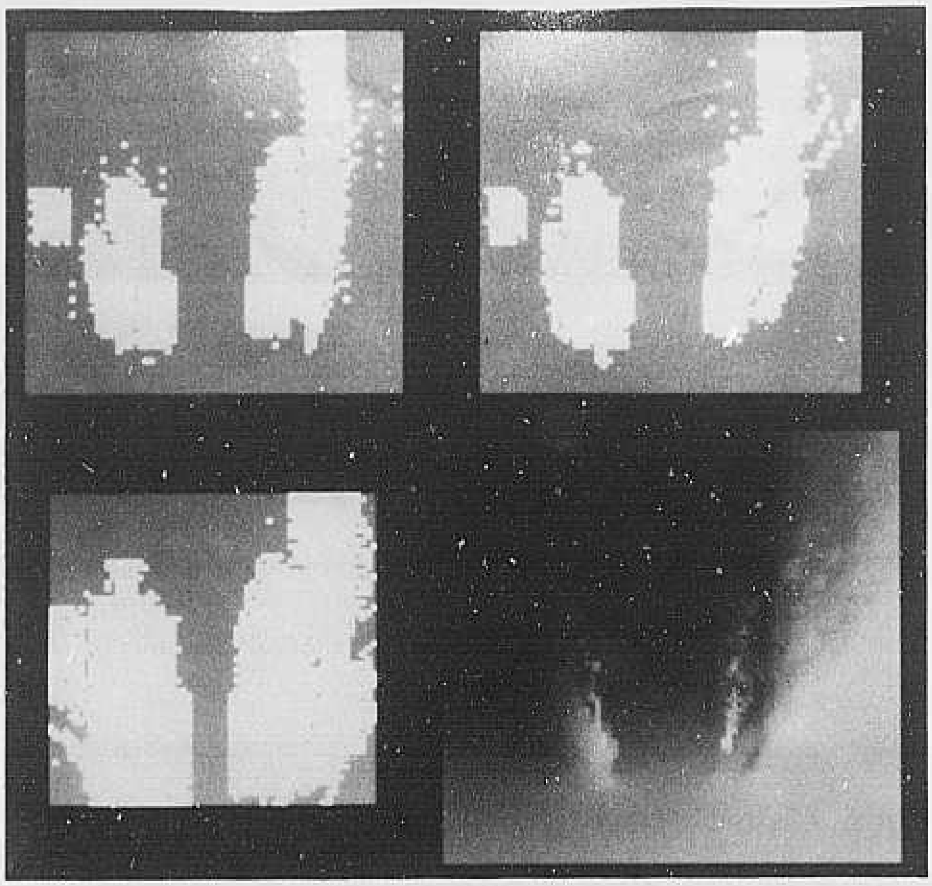


Figure 8. Traversability analysis of fence scene using vehicle model. (a) Bottom-right; Cartesian elevation map of terrain (scanner at bottom). (b) Bottom-left; large 8 wheeled vehicle (13.5 ft. by 9 ft.) at one orientation. (c) Top-left; small 8 wheeled vehicle (7 ft. by 5 ft.) at one orientation. (d) Top-right; small 4 wheeled vehicle at one orientation.

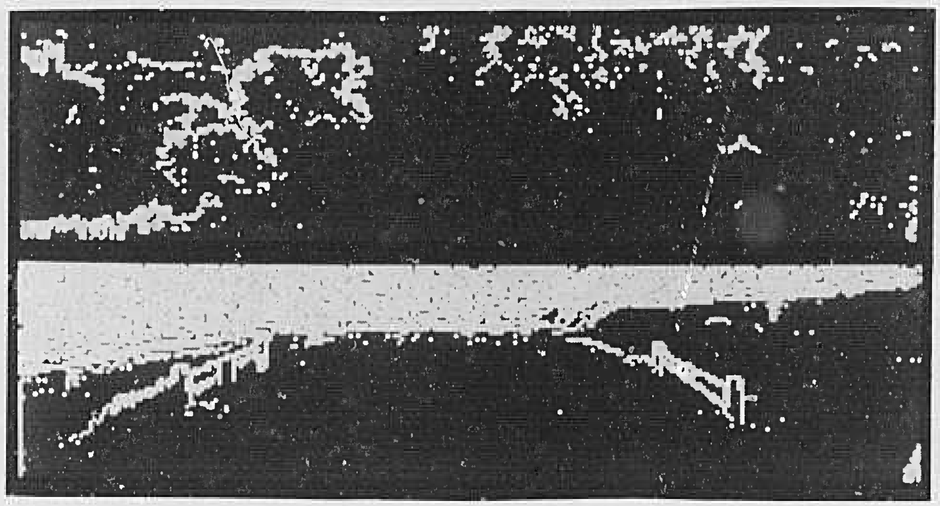


Figure 9. Discontinuities in range. White areas show occluding contour for (a) top; tree scene and (b) fence scene.

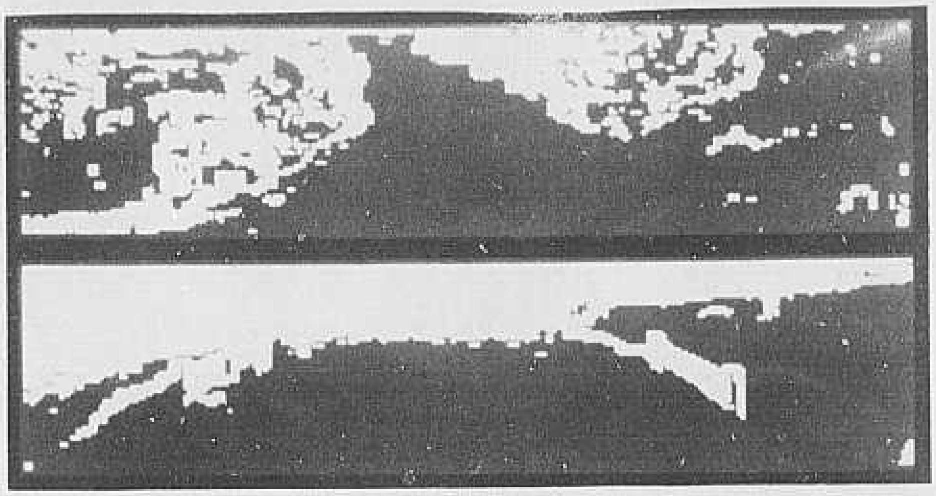


Figure 10. Discontinuities in height thresholded so white indicates at least a 1 ft. change. (a) tree scene, (b) fence scene.

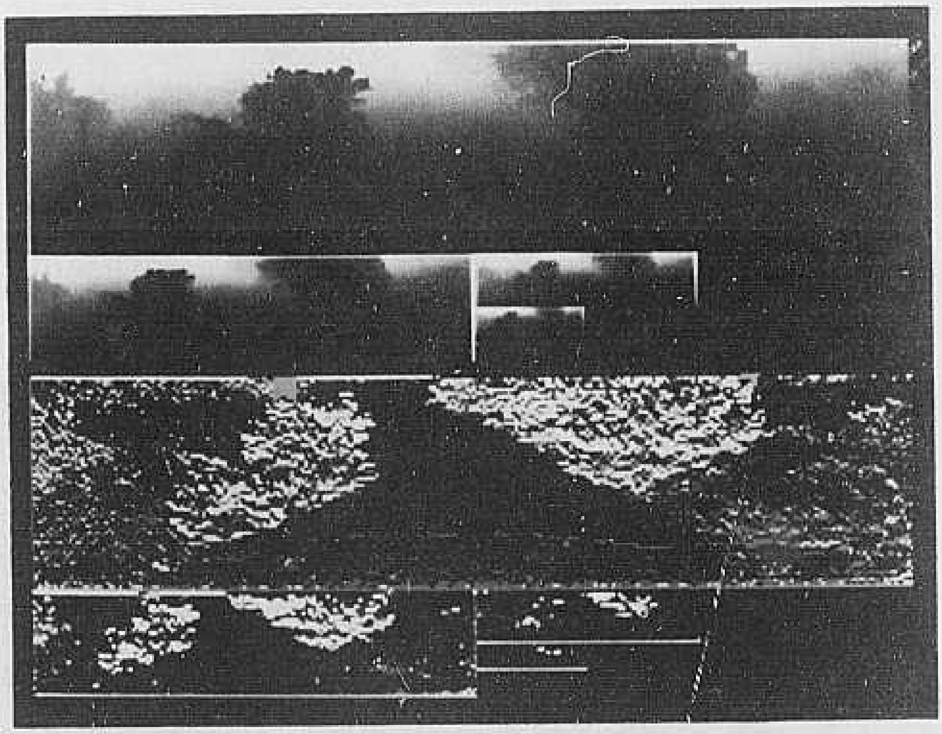


Figure 11. Multiresolution slope technique. Top four images of tree scene at resolutions of 256 x 64, 128 x 32, 64 x 16, and 32 x 8. Bottom four scenes display slope calculated using the cross product on each of the four different resolutions. Brighter areas are at steeper angles.

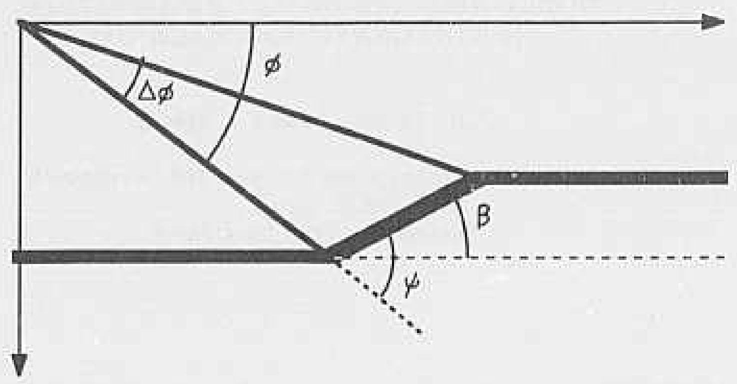


Figure 12. Geometry for radial slope calculation.

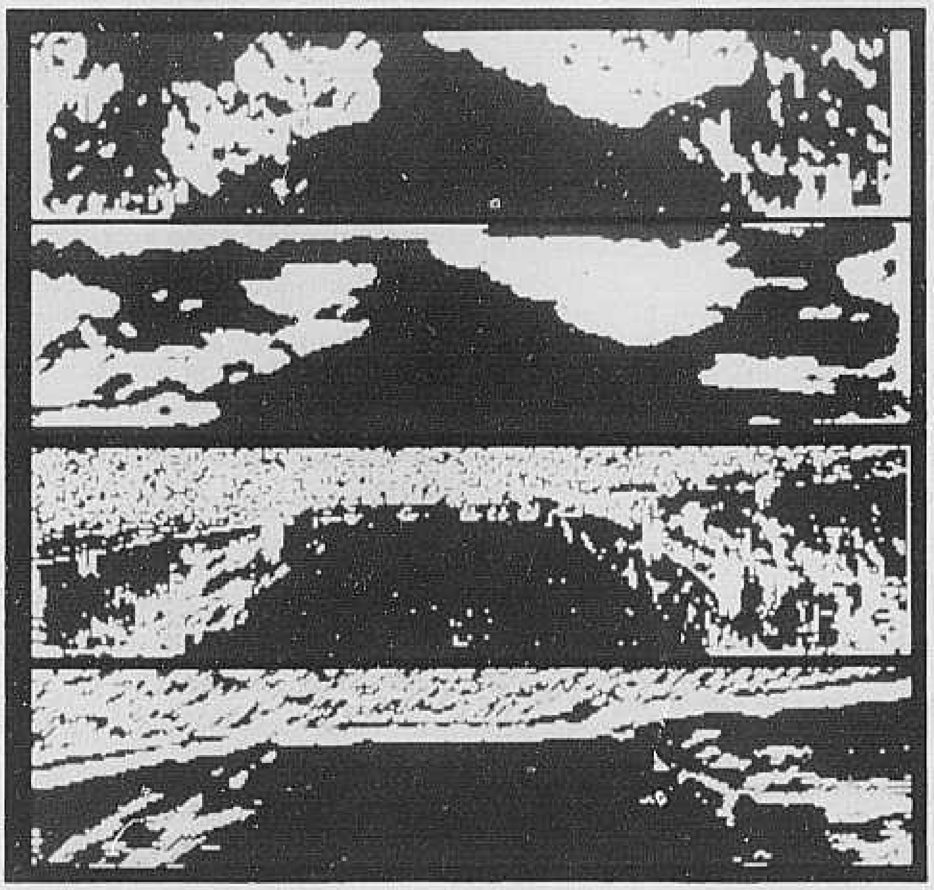


Figure 13. Slope obstacles found using cross product and radial slope techniques, thresholded at 15 degree slopes. From top to bottom (a) cross product on tree scene, (b) radial slope on tree scene, (c) cross product on fence scene, (d) radial slope on fence scene.

MODEL-DIRECTED OBJECT RECOGNITION ON THE CONNECTION MACHINE® *

D.W. Thompson and J.L. Mundy

General Electric Corporate Research and Development
P.O. Box 8
Schenectady, New York 12301

ABSTRACT

A new feature, the vertex-pair, is introduced as a basis for model-based object recognition. The vertex-pair is sufficient to define the affine transformation between a model and scene. The transform values are clustered to determine the location and orientation of an object. An implementation of the algorithm on the Connection Machine® is described.

1. INTRODUCTION

Model-Based Recognition

The use of geometric models to locate objects in visual images has been widely investigated and is growing in popularity [Besl and Jain]. Perhaps the earliest work of this nature was over two decades ago [Roberts]. In modern practice, a *model* is taken to be a precompilation of the expectations that one has about the appearance of an object in a set of image projections of the object. One widely used idea is to define a solid model of the object that can potentially generate all the possible views of the object by projection. Another idea is to define and store characteristic views [Chakravarty], which are the topologically distinct edge and vertex projections of the object.

*We gratefully acknowledge the kindness of the MIT Artificial Intelligence Laboratory for access to the Connection Machine®. The work reported here was partially funded by the DARPA Strategic Computing Program under the Army Engineer Topographic Laboratories, Contract No. DACA76-86-C-0007.

It is also common to simply store a number of intensity images of the object (or its silhouette) taken from a wide range of viewpoints. The image set is then correlated with the image to be analyzed in order to find a "best match" with one of the stored images. It is our conjecture that the geometric constraints between a three-dimensional object model and the corresponding vertex and edge features in the image provide the best foundation for the model-to-image correspondence problem.

This view is based on the observation that visual images manifest a wide variation in the intensity data associated with an object. These variations are due to such phenomena as glint, shadows, occlusion and intentional camouflage. Thus image data, as such, does not provide a reliable model feature.

On the other hand, the detection and location of object boundary curves in projection is reasonably reliable; even for small intensity contrast. This paper will discuss the problem of determining the match between a set of 3D edges and vertices, the model, and corresponding features extracted from an intensity image of the object. In particular, we define a feature grouping called the *vertex-pair* and discuss the implementation of the matching process on a highly parallel SIMD architecture, the Connection Machine.

Application Requirements

The two major military applications for object recognition are photo interpretation and target recognition. The specifications for these applications are summarized as follows:

Photo Interpretation

- Image Resolution - 8 K × 8 K Pixels
- Image Sources - Visual, Infrared and Radar
- Thruput - Perhaps 30 seconds per image
- Result - Intelligence Report

Target Recognition

- Resolution - 1 K × 1 K Pixels
- Image Sources - Infrared, Radar
- Thruput - 30 Frames/Second
- Result - Detect and Track Target

Both of these applications require the recognition of objects such as tanks, and aircraft from a wide range of viewpoints and in environments with poor contrast, low resolution and in the presence of occlusion and camouflage. It is our view that the use of an explicit three-dimensional model is necessary to provide enough context to support the recognition of objects in such complex scenes. The current performance of edge detectors and line segmentation algorithms leads to short and disconnected line fragments. A model allows the segments to be grouped and assigned to projections of the model with good reliability even in the case of occlusion and noise.

It is also important to provide sufficient thruput to support the applications described earlier. A major factor is the lack of reliable, significant features in an image segmentation. Occlusion, shadows, and poor contrast can break up or eliminate any major object features that might be used to direct a focussed search for object matching. We consider that any edge fragment or vertex in the segmentation is a feasible candidate for assignment to a corresponding edge or vertex in the model. As a consequence, we are forced to try all combinations of assignments. In this paper we introduce a feature grouping that requires only two vertices, as well as two edges associated with one of the vertices. This pairwise grouping leads to an algorithmic complexity of N^2 in the number of scene vertices, N.

While this complexity is not exponential, the amount of computation required for a practical application images is beyond the thruput of conventional architectures, e.g., the Lisp Machine. Consequently, a major focus of this paper is the use of the Connection Machine (TM), which is a fine-grained SIMD architecture. We believe that this type of architecture is well suited to the type of matching algorithm described below. For example, it is not difficult to generate scenes of sufficient complexity to require 1 hour of match time on the Lisp Machine for each model feature. The current implementation on the Connection Machine (TM) with 8 K processors requires as little as 45 seconds per model feature.

2. THE MATCHING ALGORITHM

In this section we describe the process for generating an affine transform from a correspondence between vertex-pairs in a model and those derived from the segmentation of an intensity image. In addition, an algorithm for clustering the transform values to find an appropriate match between the model and scene is also presented.

The Model Vertex-Pair

In the work described here, we assume that a three-dimensional polyhedral model is available for the set of objects to be recognized. The use of polyhedral models does not restrict the object being viewed to be a polyhedron, but only that the significant edges and vertices that appear in an image of the object be accurately predicted from an affine projection of the model. In fact, the complete properties of a polyhedron are not required in the matching process. The two main requirements are

- Model vertices and edges are in correct geometric correspondence
- Sufficient information is available to determine visibility of edges and vertices from all viewpoints.

For example, it is not, strictly speaking, necessary to have the faces of the polyhedron lie in planar surfaces as long as the face boundaries are in correct position and visibility calculations provide correct constraints.

We use a restricted form of perspective image formation, which is the *affine* transform [Roberts]. The affine transform preserves the parallel relation between two lines in projection. The full perspective transformation reduces to the affine approximation when the depth variations within an object are small compared with the distance of the object from the camera [Thompson and Mundy]. This condition is met in a wide variety of object recognition tasks for military and industrial applications.

It can be shown that the six parameters of the affine transform, three translations and three rotations, can be derived from a group of model edges and vertices called the *vertex-pair.* The geometric arrangement of the vertex-pair is illustrated in Figure 2.1. The vertex associated with the two edges is called the base vertex. The line defined between the two vertices is called the spine. Note that the spine does not actually have to correspond to an edge in the model or in the image. The angles between each edge and the spine are defined as α and β . The rotation angles about the x,y,z axes are ϕ,ψ,ζ , respectively. These rotations define the orientation of the model vertex-pair with respect to the image plane coordinate system.

The Scene Vertex-Pair

The scene vertex-pairs are derived by a standard image segmentation approach. The Canny [3] edge detector is used to locate object boundary points. Briefly, this edge detector applies a convolution for the first spatial derivative at each point in the image. The operator is applied in four orientations at 45-degree intervals. The direction with maximum first derivative magnitude is chosen to compute the second directional derivative. A potential edge point is defined at the zero crossing of the second derivative.

In our experiments, we do not use a threshold on first derivative magnitude to eliminate edge points, except for the unavoidable quantization of the integer calculations. That is, a threshold of 1 is used, which includes edges of any non-zero magnitude. We have adopted this policy since valid edge events in our experimental image set can occur over the full dynamic range of strength values.

The edge points are collected into connected regions and thinned into one-dimensional curves. The one-dimensional curves are approximated by a connected sequence of linear segments. The breakpoints between these segments are determined by points of maximum curvature on the curve [Assada and Brady]. We determine these locations by decomposing the curve into two functions x(t), y(t), where t is a parameter running along the curve.

These functions are smoothed at a series of spatial scales by recursively applying a 3-wide Gaussian operator to each function. The locations of extremal tangent angle change along both curves are marked at each scale. The maximum angle change is determined by the zero crossing in a second derivative operator applied to each function. A location is taken to be a segment breakpoint if the zero crossings in either x(t) or y(t) occur at many scales and the angle change is considered to be significant [Assada and Brady]. We have used a threshold of 0.2 radians to define a significant curvature maximum.

Line segments that are less than about 10 pixels in length are eliminated since they are not considered long enough to define an accurate orientation in the image view plane. For example, a variation of plus or minus one pixel at each end of an edge five pixels long corresponds to an orientation variation of more than 20 degrees. Vertices are then formed by intersecting the remaining edges. The edges are allowed to be extended by as much as one half their length at each end to discover potential intersections.

The results of the segmentation process just described is now applied to the image of a C130 cargo plane, which is shown in Figure 2.2. This image was obtained from an altitude of about 500 feet. The resulting segmentation is shown in Figure 2.3. The vertices are shown as small circles; the segmentation produced 90 vertices.

All of the vertices in the scene are grouped pair-wise to generate all possible vertex-pairs. For each pair, additional instances are generated to account for edge orientation and cases where a vertex may represent the intersection of more than two edges.

We do not eliminate any pairs from consideration at this point, since there is no good basis for deciding that an edge or vertex cannot be part of the projection of the object in question. For example, edge segment length is not a strong criterion since boundary curves are easily broken up by occlusion and camouflage. Also we assume that intensity or color are not good indicators of object class. We rely entirely on the geometric constraints imposed by the model.

Computing and Clustering the Transform

We now have two sets of vertex-pairs, model vertex pairs which are selected from a three-dimensional polyhedral model and those generated from the image as just described. The vertex-pair defines enough constraints so that all of the parameters of an affine transformation between a model vertex-pair and a scene vertex-pair can be computed. The computation is summarized in reference to Figure 2.1 as

- The projected angles between the spine and each edge, α and β determine the tip and tilt angles of the model coordinate frame, ϕ and ψ with respect to the image plane.
- The orientation of the projected spine with respect to the x axis of the image plane determines the rotation about the z axis, ζ.
- The translation between the projected base vertex of the model vertex-pair and the actual image location of the scene vertex-pair gives two of the translational degrees of freedom.
- The ratio of the length of the projected model spine and the actual image spine length gives the affine scale factor.

These steps nominally produce a point in six-dimensional transform parameter space for each correspondence between a model vertex-pair and a scene vertex pair. Actually more transform values may be produced from a correspondence due to degeneracy or multiple solution in the mapping between $< \alpha , \beta >$ and $< \phi , \psi >$ [Thompson and Mundy].

We define a potential object match by a tight cluster in transform space [Tanaka et al., Silberberg et al.]. The cluster is detected by a combination of histogramming and a form of the nearest neighbor clustering algorithm. The transformation instances are counted into a two-dimensional histogram based on values of $< \phi , \psi >$. If a bin contains at least a specified number of points, the contents of the bin are projected onto a histogram on values of ζ. Increments of five degrees are used in both histograms.

Any set of transform values that pass through the angle sieve is further tested by a nearest neighbor clustering in translation-scale space. A first point is selected to serve as the center of the cluster and then other points in the set are eliminated if they exceed tolerances on translation and scale with respect to the center value. If a point passes the test, it is added to the cluster and the center is recomputed.

Results

The model of the C130 was obtained from *Jane's Aircraft of the World,* which is a standard reference for aircraft specifications. A fairly complete model is shown in Figure 2.4. This model was created using conventional CAD tools. A simplified model derived from this data is shown in Figure 2.5. We have observed that most of the significant edges and vertices that are robustly produced in images of the aircraft are accounted for by this simpler model.

The simple model is matched into the image and an example of a correspondence is shown in Figure 2.6. The model is projected into the scene according to the transformation determined from the vertex-pair match. A projection according to the average transform computed over five vertex-pair assignments is shown in Figure 2.7. The angular errors are about five degrees and translation errors amount to about 10% of the object dimension in the image. Scale variations are also on the order of 10%.

The Lisp Machine implementation timing has been briefly studied. The experiments were run on a Symbolics Inc. 3600 Lisp Machine with 2 Mwords of memory and no floating point accelerator. The preliminary timing for the C130 image is as follows:

- Image Segmentation - 30 min (455 x 204 pixels)
- Vertex Pair Generation - .1 sec per scene vertex-pair (26,000 vertex-pairs were produced)
- Clustering - .5 sec per scene vertex-pair per model vertex-pair.

It is emphasized that these results are very preliminary and are based only on the C130 experiment. We include them only to give a rough comparison with the Connection Machine(TM) implementation that we now describe.

3. THE CONNECTION MACHINE IMPLEMENTATION

Overview of the Connection Machine

The Thinking Machines Inc. Connection Machine®, shown in Figure 3.1, is a SIMD architecture composed of between 8K and 64K 1-bit processors, each with about 4K of memory in the current version. The user interface is provided by a front end machine, typically a Symbolics Inc. Lisp Machine. The processors are connected by a grid for fast local communications and a router for arbitrary interprocessor communication.

A single stack is maintained for the entire processor array, so that each processor is expected to have the same number of bits allocated for the same purposes at all times during its operation. Instruction execution occurs simultaneously at every processor in a subset of the processor array. The subset, called the selected set, is determined by some parallel binary variable. The binary variable is true (non-nil) in the members of the selected set, which are called the active processors.

Processors may he addressed either relative to each other or in absolute coordinates. Either mode allows addressing in either one- or two-dimensional addressing schemes. A set of scan operations is available to effect high-speed operations such as copy and max over a series of processors. The operational characteristics of the CM (SIMD, limited processor memory, common stack for all processors) constrain the implementation of the matching algorithm. In particular, individual processors may only store a fixed amount of data, and each active processor must have the same data allocation bit-wise.

The limited I/O bandwidth between the CM and its Lisp Machine host makes it beneficial to avoid such I/O as much as possible. Inter-processor communication using the router should be minimized, especially where the possibility of hot points (processors receiving many simultaneous read or write requests) exists. It is best to do as much computation as possible within a processor.

Edge, Line, and Vertex Detection

Edge detection on the CM uses a parallel implementation of the Canny [3] edge detector developed by Todd Cass at the MIT AI Lab. Line and vertex detection are similar to the serial implementation, but pixel operations are carried out in parallel. First, edge pixels produced by the Canny edge detector are thinned and then sorted into connected components by labeling them with the maximum of the one-dimensional addresses of their two endpoints (edges without endpoints are labeled by their maximum processor address). Next, edge pixel locations are separated into x and y dimensions and each dimension is considered as a separate 1D curve.

Curvature in each dimension is computed by applying a 1×3 curvature kernel [-2, 4 -2] to each location curve at each edge pixel. To locate local maxima of curvature, a small Gaussian kernel is repeatedly applied to the curvature data. Each pixel keeps a count of the number of times it has high curvature and a zero crossing of the second derivative of curvature in either x or y during the smoothing. After a number of smoothing passes, those pixels that exhibited high curvature at a number of spatial scales are marked as breakpoints. Lines are segments which sequentially join the breakpoints and endpoints derived from a single connected component.

Vertices are extracted from the set of edges by extending edges at each end and arithmetically intersecting each extended edge with every other extended edge. The set of extended edges is placed along the first row and the first column of processors in the two-dimensional addressing scheme. The edge data is scanned into the processor array by rows and columns so each processor holds two edges. Those processors whose row address is less than their column address (to avoid duplicating intersections or self-intersections) then compute the intersection of the two edges they hold and mark themselves if an intersection occurred. Scene vertex-pairs are generated from the resulting edges and vertices, and serve to completely represent the scene during all further processing.

Mapping the Matching Problem Onto the CM

To compute a set relation between a small set and a much larger set on the CM, it is natural to configure the environment so that each element of the larger set is represented by an individual processor and every element of the smaller set exists in every processor. This is useful when size of the vertical set approaches the number of processors and the size of the horizontal set is small enough that there is sufficient memory to represent the value of each element at every processor.

The recognition algorithm described here is such a set operation, as the number of model features is generally quite small, and the number of scene features is large, comparable to the number of processors. Therefore an active processor will contain structures describing one scene feature and all model features. There will be as many active processors during matching as there are scene vertex pairs. In accordance with the constraints of the CM discussed above, these structures must be compact, to avoid stack overflow, and should contain all relevant information, to avoid unnecessary I/O.

There may exist a number of different transformations which cause a model vertex-pair to project to the angles of a particular scene vertex-pair. Because the CM must allocate the same number of bits at each processor, all the possible transformations cannot be stored in an individual CM processor. In other words, an individual processor cannot completely describe the possible relations between a model feature and a scene feature.

Furthermore, the set of transformations mapping a model vertex-pair to a scene vertex-pair are votes in transform space. But if they have similar values, a single pairing of a model vertex-pair to a scene vertex-pair may produce several votes in a single transform bucket. The simple population of a cluster is not a good indication of its significance. Each vote must be associated with the identities of the features that generated it, so that multiple votes by the same model-scene pair can be accounted for. This requirement, which was incorporated in the serial implementation, is impractical to implement on the CM because of the common stack and limited local memory; consequently, the CM cannot reasonably be configured to represent the complete transform space in such a straightforward manner.

It is possible to represent a subspace of the transform space in the CM by comparing the scene to the model at a single rotation. Each rotation of the model about the X and Y axes of the image plane (ϕ and ψ, respectively) produces a single unique projection. Therefore, each model vertex-pair will have a single projection, which can then be compared with the set of scene vertex-pairs. This produces a single transform for each pair of model vertex-pair and scene vertex-pair. It is then possible to store the single transform since the same amount of memory is allocated at each processor. Furthermore, since each model vertex-pair only votes for one transform when it is paired with a scene vertex-pair, transform clusters can be selected based on their bucket population alone.

The entire matching process is executed by cycling through all possible values of rotation. At each rotation, the active processors, those which represent scene vertex-pairs, compare themselves in parallel to each model vertex pair in turn. When the projection of the model vertex-pair is similar to a scene vertex-pair, a vote is made in transform space for the transform which produced the match. Clusters of transforms are grown to locate transforms which occurred frequently, which may indicate the presence of instances of the model in the scene.

Data Structures

There are four data structures used in the CM implementation of matching. Model vertex-pairs are represented in the CM by their set of possible projections. A projection of a model vertex-pair is described by 6 values:

- α: the projected angle between the spine and w1 in degrees
- β: the projected angle between the spine and w2 in degrees
- γ : the projected angle of the spine in degrees
- x: The projected X location of the base vertex
- y: The projected Y location of the base vertex
- Scale: The projected length of the spine

The set of possible projections is represented as an array called an $\alpha-\beta$ map. Element (i, j) of an $\alpha-\beta$ map contains the parameters of the projected model vertex-pair after rotation i degrees about the world X axis and j degrees about the world Y axis. Rotations are represented in 5 degree increments, giving 36 $\times$ 72 entries in the $\alpha-\beta$ map. Using the two-dimensional addressing scheme of the CM, an $\alpha-\beta$ map is stored in the CM so that array element (i, j) is stored in processor (i, j) (Figure 3.2). Each model vertex-pair is represented in the CM by a separate $\alpha-\beta$ map.

Each element in the $\alpha-\beta$ map holds 7 values:

- α (9 bits)
- β (9 bits)
- γ (9 bits)
- x (depends on image size: 9 bits for 256 wide)
- y (depends on image size: 9 bits for 256 high)
- Scale (typically a 32 bit float)
- A single bit tag field

The total is 78 bits for a 256 $\times$ 256 image.

Scene vertex-pairs are represented in the CM by the same 7 values as the model vertex-pair (α, β, γ, x, y, scale and a tag bit). For the scene vertex-pair, these values are constants. Therefore each scene vertex-pair can be represented in one 78-bit field. To store scene vertex-pairs, 78 bits are allocated in every processor in the CM. Each processor is loaded with the data from one vertex-pair, for as many vertex-pairs as exist. The tag bit marks whether a processor is actually loaded with a scene vertex-pair or not, and is used to define the selected set of scene vertex-pairs.

Transform clusters hold the average values of a set of similar transforms. The transformation is described by 6 values:

- ϕ : rotation about the world X axis (same as image X axis)
- ψ : rotation about the world Y axis (same as image Y axis)
- ζ : rotation about the world Z axis (viewing axis)
- Dx : translation in X
- Dy : translation in Y
- scale-ratio : scale factor, corresponds to distance of object from viewplane

Like the scene vertex-pair, a single processor stores the value of a single cluster. Memory for one cluster is allocated in every processor. A tag bit is used to mark whether a processor contains a cluster or not. Each cluster processor holds 8 values:

- ϕ (9 bits)
- ψ (9 bits)
- ζ (9 bits)
- Dx (depends on image size: 9 bits for 256 wide)
- Dy (depends on image size: 9 bits for 256 wide)
- Scale-ratio (typically a 32 bit float)
- Population (number of transforms in cluster, size depends on number of model vertex-pairs, typically 5 bits)
- A single bit tag field

The total size is 83 bits for 256 $\times$ 256 image.

The next data structure is used to hold copies of single elements of the $\alpha-\beta$ map. An instance of this structure, called a match field, exists for each model vertex-pair. The match field allows each processor, which represents a scene vertex-pair, to have its own copy of an entry in an $\alpha-\beta$ map. Therefore, rather than being the size of the $\alpha-\beta$ map, the match field is active only in processors that represent scene vertex-pairs.

Values from a single element of the $\alpha-\beta$ map, corresponding to a single (ϕ, ψ) rotation, are copied into every element of the match field. The elements are then replaced by transform values by comparing each entry with corresponding scene vertex-pair properties in each processor. The match field has the same entries as the $\alpha-\beta$ maps: three rotations, two translations, a scale factor (translation along the viewing axis) and a tag bit.

Each model vertex-pair therefore requires 146 bits (78 for the $\alpha-\beta$ map and 78 for the match field). The scene vertex-pair structure requires 78 bits. The cluster structure has 78 bits (identical to the match field).

Since there is only one scene vertex-pair structure and one cluster structure, each processor must allocate 151 bits plus 146 bits for each model vertex-pair (Figure 3.3). More than 10 model vertex-pairs are likely to overflow the limited memory on each processor in the current version of the machine. The data from the $\alpha-\beta$ map could be input at runtime from the Lisp Machine front end, to eliminate the need for the $\alpha-\beta$ maps altogether, at the expense of some extra I/O.

Determining the Transformations

In order to derive transforms between a model vertex-pair and the set of scene vertex-pairs, a single set of model values from the $\alpha-\beta$ map is scanned into the match field, so that the match field contains the projected parameters of the model resulting from one (ϕ, ψ) in every active processor (every processor representing a scene vertex-pair) at once (Figure 3.4).

Each active processor then tests to see if the scene α and β it represents are within a narrow threshold of the projected α and β from the match field. Those processors that are not near in value are made inactive. The set of active processors then compute the remaining degrees of freedom (rotation about the viewing axis (ζ), X and Y translation in the image plane and scale) by a comparison between the scene values and model values in the match field. These values are stored back into the match field for cluster generation.

Growing Transform Clusters

Transformation clustering occurs when a set of model vertex-pairs require the same (within error tolerance) transformations to project to a set of scene vertex-pairs.

The main problem with forming transform clusters is that a given model vertex-pair may project to approximately the same image angles (α and β) at several slightly different rotations (ϕ and ψ). This may produce several votes for the same cluster by a single mapping of a model vertex-pair to a scene vertex-pair, which cannot be allowed. To insure that a single model-scene mapping produces only one vote in any cluster, clustering is done separately for each value of ψ and ϕ. Clusters generated at each ψ and ϕ are considered complete and correspond to solutions (object instances) if they contain enough votes.

Clustering starts when a set of transform values is computed as described above, comparing the first model vertex-pair to the set of scene vertex-pairs at a single (ψ, ϕ). Each active processor (each processor which computed a plausible transform) then starts a transform cluster with the transform values it computed. Transforms are then derived for each successive model vertex-pair at the same (ψ, ϕ).

After each transform derivation, each active processor either adds its vote to an existing transform cluster or creates a new one if none are reasonably close in value. An active processor may vote for only one cluster, and no two processors will vote for the same cluster since their transforms must be significantly different.

Every processor with an active transform must query each cluster processor to check if the transform is within bounds of the cluster. Each transform processor simultaneously queries the cluster processors in a circular order. The address of the first cluster processor queried by a transform processor is computed to minimize the number of simultaneous queries to any one processor (Figure 3.5).

To create a new cluster a new processor is made active as a cluster and its cluster values are set by the values in the match field of the processor creating the cluster. Voting for an existing cluster consists of incrementing the vote count of the cluster and modifying its values, which are the averages of the individual values from the member transforms.

Timing

For each (ϕ, ψ), each model vertex-pair is compared in turn to the set of scene vertex-pairs. This comparison involves scanning the projected values of the model vertex-pair from the $\alpha-\beta$ map into every element of the match field, computing the transforms at every scene model vertex pair which is similar to the projected model vertex-pair, and clustering the transform values.

Scanning an element of the $\alpha-\beta$ map of a model vertex-pair into every element of a match field takes 5 msec of CM time, 10 msec elapsed time. Computing the transform values for a single model vertex-pair at a single rotation takes 8 msec of CM time, 32 msec elapsed time. Clustering is the most time consuming step. Creating new clusters takes 1 msec of CM time, 4 msec of elapsed time. Adding votes to clusters is done as many times as there are clusters, while each transform processor queries each cluster in turn. The number of clusters is generally related to the number of scene vertex-pairs: the more scene vertex-pairs there are, the more likely false transforms will be computed. However, it is unlikely that false transforms will cluster. Adding votes to clusters and recomputing cluster values consumes 18 cm msec, 45 elapsed msec.

If we assume 5 votes are added per rotation, each model vertex-pair at each rotation will require approximately 300 msec elapsed time. That is, approximately 13 minutes per model vertex-pair. In practice, many $\phi-\psi$ rotations produce degenerate projections or projections that do not correspond to any scene vertex-pair, and so are ignored. Also, the visibility of a model vertex-pair can eliminate certain rotations from consideration a priori. A simple scene can take as little as 45 seconds elapsed time per model vertex-pair.

By comparison, the Lisp Machine implementation requires about 0.5 seconds for the corresponding calculations. However, the CM can execute in parallel over all scene vertex-pairs. A typical scene might have as many as 10^5 vertex-pairs. Assuming that the CM has enough processors to allocate a processor for each scene vertex-pair, then the speedup is on the order of 10^3. More study will be needed to verify this conjecture, since the current operations on cluster formation are serial by cluster. A more complex scene will take longer to do the clustering then the current observed result. On the other hand, we expect to improve this section of the implementation shortly.

Clustering of transforms is the area where performance improvements would be the most profitable. It is possible to sort the clusters on one of the transform parameters, such as ζ rotation. The location of relevant clusters during the voting process can then be done in logarithmic rather than linear time in terms of the number of defined clusters.

REFERENCES

[1] Asada, H. and Brady, M., "The Curvature Primal Sketch," *Proc. IEEE Workshop on Computer Vision: Representation and Control*, 1984.

[2] Besl, P. and Jain, R., "Three Dimensional Object Recognition," *Computing Surveys 17*, (1), March 1985.

[3] Canny, J., "Finding Edges and Lines in Images," MIT Artificial Intelligence Laboratory Report, AI-TR-720.

[4] Chakravarty, I. "The Use of Characteristic Views as a Basis for the Recognition of Three-Dimensional Objects," PhD Thesis, Rensselaer Polytechnic Institute, 1982.

[5] Roberts,L. G., "Machine Perception of Three Dimen-

sional Solids," *Optical and Electro-optical Information Processing,* J.T. Tippett et al., eds., MIT Press, Cambridge Mass., 1965.

[6] Silberberg, T., Harwood, D. and Davis, L., "Object Recognition Using Oriented Model Points," *Computer Vision, Graphics and Image Processing 35,* 1986.

[7] Tanaka, H.,Ballard, D.,Tsuji, S., and Curtis, M., "Parallel Polyhedral Shape Recognition," *Proc. CVPR,* 1985.

[8] Thompson, D.W. and Mundy, J.L., "Three Dimensional Model Matching from an Unconstrained Viewpoint," *Proc. IEEE Robotics and Automation,* 1987.

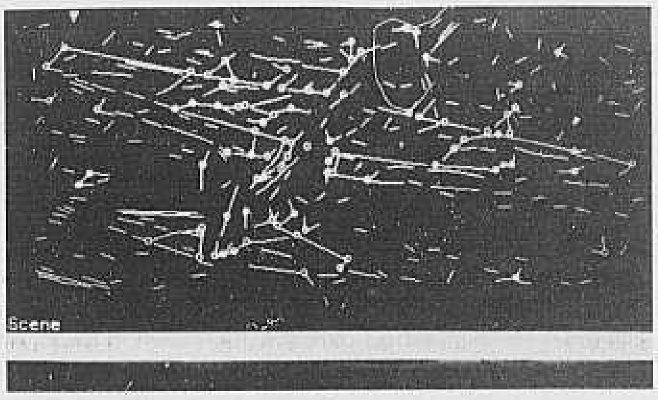


Figure 2.3. The segmentation of the image in Figure 2.1. The vertices are shown as small circles.

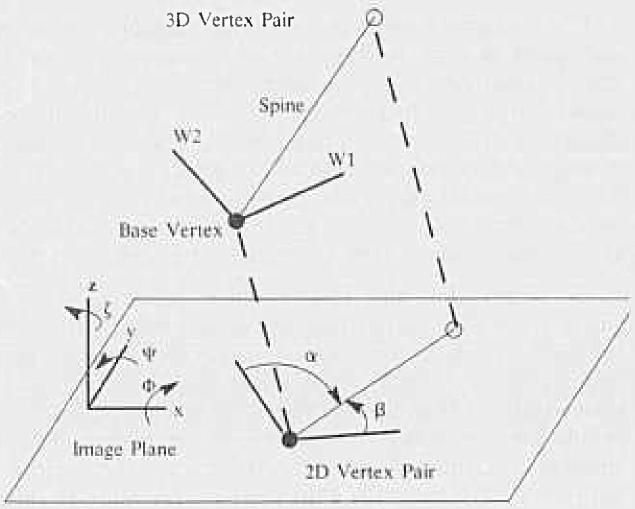


Figure 2.1. The vertex-pair geometry.

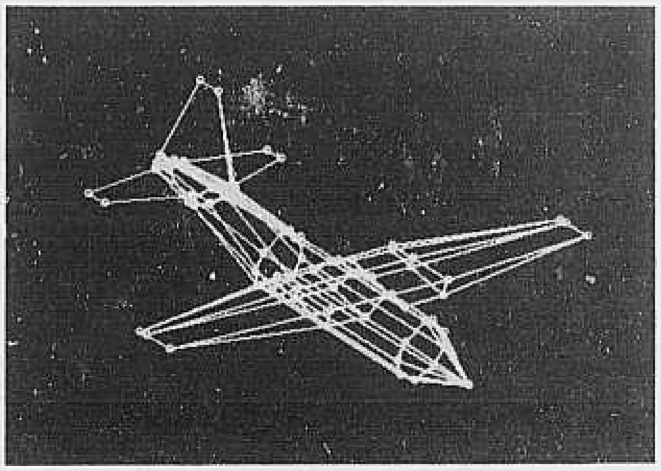


Figure 2.4. A polyhedral model for the C130.

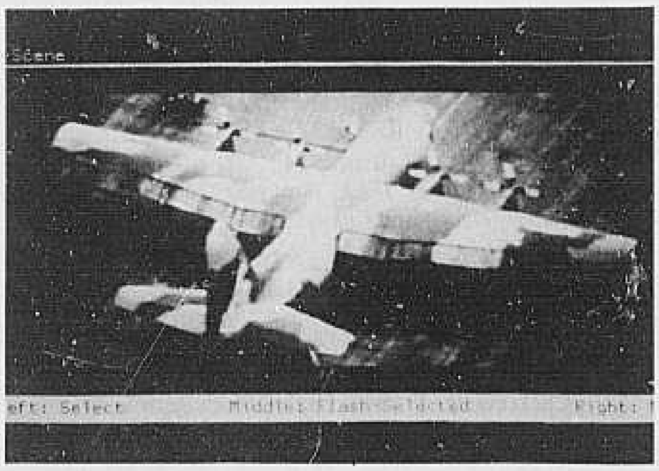


Figure 2.2. An aerial photograph of a C130 cargo transport plane.

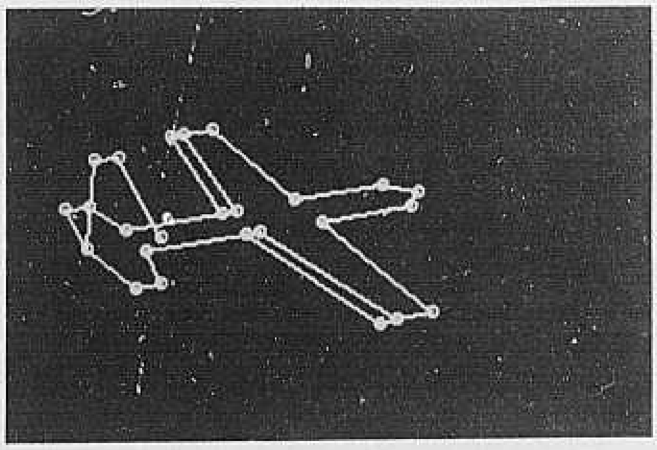


Figure 2.5. A simplified model of the C130.

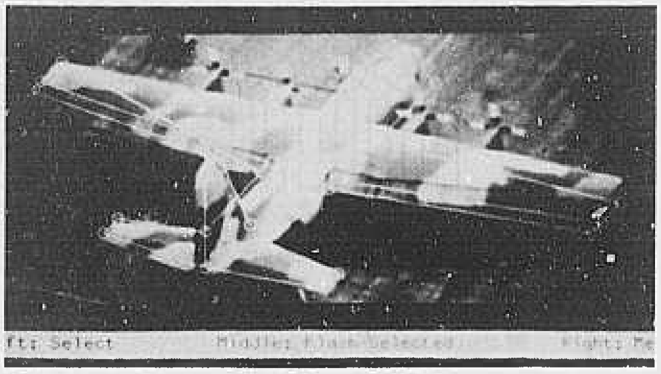


Figure 2.6. A match with one of the vertex-pairs of the model. One vertex is the left front wing tip the other is at the intersection of the left wing with the fuselage. This vertex pair is indicated with the thick white line.

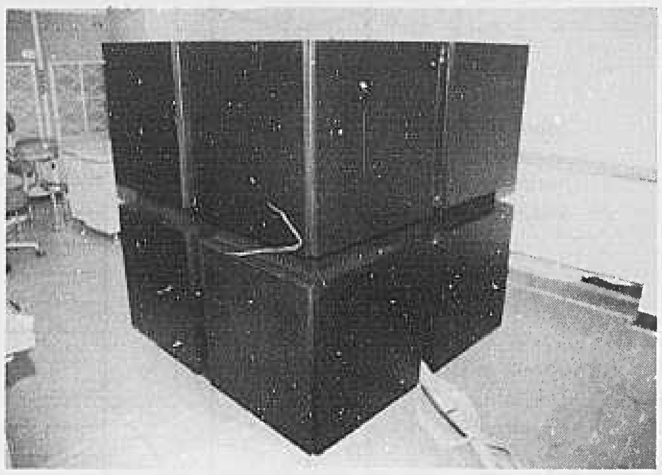


Figure 3.1. The Thinking Machines, Inc., Connection Machine®.

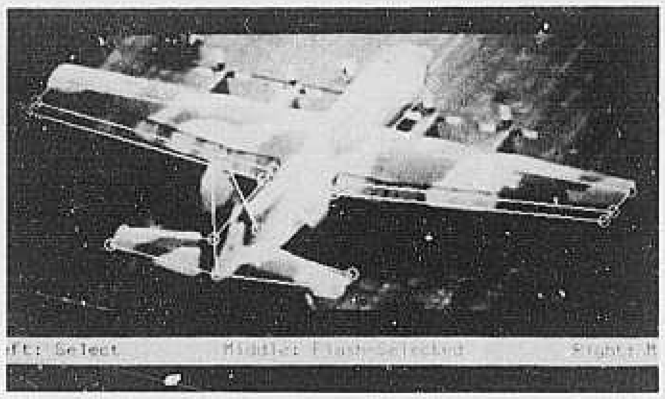


Figure 2.7. A composite match for a cluster of five vertex-pairs. The transformation indicated by the overlay is the average of the transform values over the cluster.

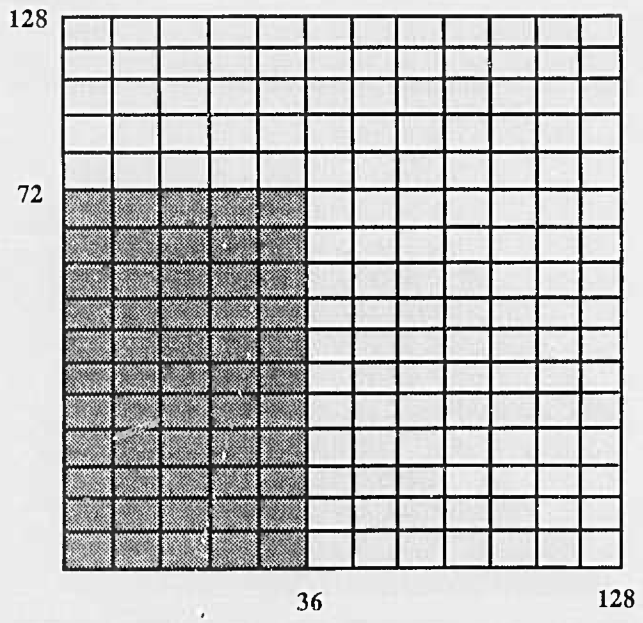


Figure 3.2. Storage of a model alpha-beta map by processor address in a 16k Connection Machine®.

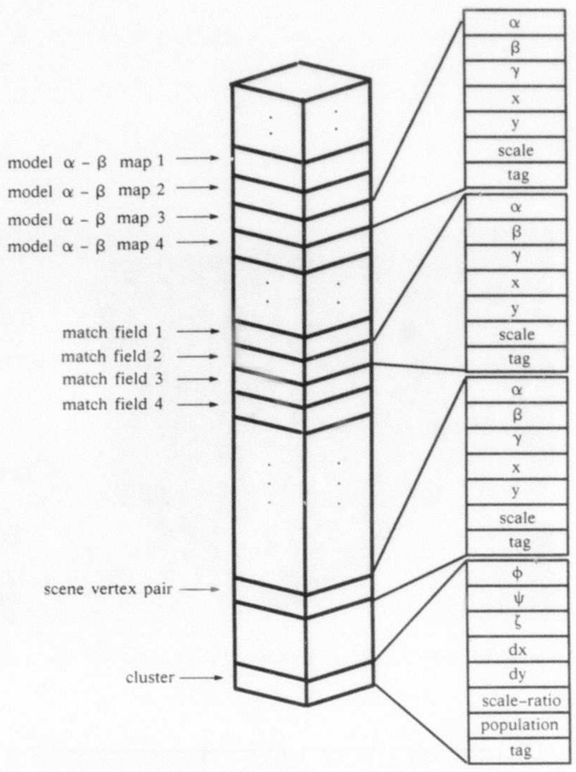


Figure 3.3. Arrangement of data structures in the memory of a single CM processor.

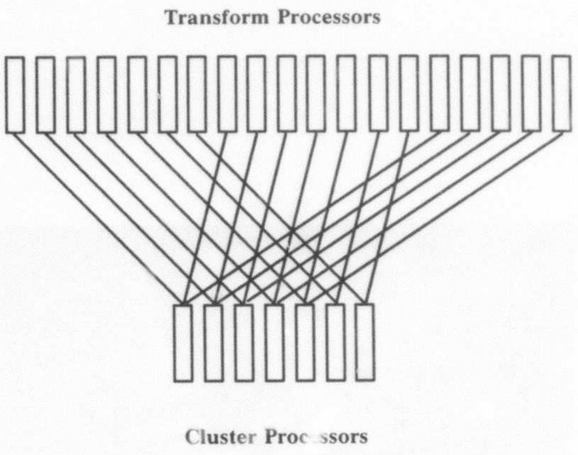


Figure 3.5. Transform processors query cluster processors with minimum multiple requests.

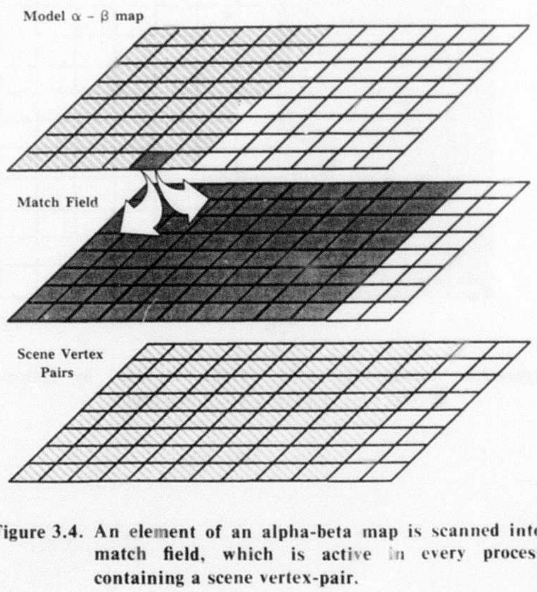


Figure 3.4. An element of an alpha-beta map is scanned into a match field, which is active in every processor containing a scene vertex-pair.

ENVIRONMENTAL MODELING AND RECOGNITION FOR AN AUTONOMOUS LAND VEHICLE

Daryl T. Lawton, Tod S. Levitt, Christopher C. McConnell,
Philip C. Nelson, Jay Glicksman

Advanced Decision Systems, Mountain View, California 94040

ABSTRACT

We present an architecture for object modeling and recognition for an autonomous land vehicle. Examples of objects of interest include terrain features, fields, roads, horizon features, trees, etc. The architecture is organized around a set of data bases for generic object models and perceptual structures, temporary memory for the instantiation of object and relational hypotheses, and a long term memory for storing stable hypotheses that are affixed to the terrain representation. Multiple inference processes operate over these databases. We describe these particular components: the perceptual structure database, the grouping processes that operate over this, schemas, and the long term terrain database. We conclude with a processing example that matches predictions from the long term terrain model to imagery, extracts significant perceptual structures for consideration as potential landmarks, and extracts a relational structure to update the long term terrain database.

1. INTRODUCTION

Terrain and object models for autonomous land vehicles (ALVs) are required for a wide range of applications including route and tactical planning, location verification through the recognition of terrain features and objects, and acquiring new information about the environment as it is explored. The following lists important criteria for terrain and object modeling capabilities.

Descriptive Adequacy: The modeling technique should be capable of describing the objects and situations in the environment necessary for the vehicle to function. This includes representing natural as well as man-made objects. It should be a consistent representation that supports modular system development and uniform inference procedures that can operate over different types of objects at different levels of detail. Uniform shape, object subpart and surface attribute affixments are necessary to do this.

Recognition Adequacy: Much of the activity of an ALV is concerned with determining where it is and what is around it. Terrain models should be manipulable for determining the sensor-based appearances of world objects and for controlling recognition processing. This involves the formation of general predictions of sensor derived features from the terrain model. Such predictions will often be uncertain and qualitative due to incomplete prior knowledge of the terrain.

Handling Uncertainty: The existence and exact environmental location of objects will often not be known with complete certainty. Locations will often be determined relative to other known locations and not with respect to a globally consistent terrain map. This is true, for instance, when the sensor displacement parameters are not well determined. It is necessary to represent this uncertainty explicitly in the terrain model so incrementally acquired information can be used for disambiguation.

Learning: A vehicle will learn about the environment as it moves through it. Associating new information with the terrain representation should be straightforward. This is difficult to do, for example, by changing values in a raw elevation array. Types of information to be affixed to the terrain representation include newly discovered objects, details of expected objects, and the processing used in object recognition.

Fusion of Information: The ALV must build a consistent environmental model over time from different sensors. As an object is approached, its image appearance and scale will change considerably, yet it has to be recognized as the same object, with newly acquired information associated with the unique instance of the general object type. In a typical situation, a distant dark terrain patch will be partially recognized based upon distinctive visual characteristics, but may be either a building or a road segment. As it is approached, its image appearance changes considerably, making disambiguation possible. This requires the representation of multiple hypotheses, each formated with respect to the properties of the potential world objects. The structure of the object description should direct the accumulation of information.

A further consideration in developing and evaluating terrain modeling capabilities is that there is not a single ALV. Instead, there are a wide range of autonomous vehicles, indexed by a diverse range of active and passive sensors and assumptions about a priori data. There is a continuum from systems having a complete initial model of the terrain and perfect sensors to those with no a priori model, and highly imperfect sensors. For example, a robot with no a priori data and only an unstabilized optical sensor will probably model the environment in terms of a sequence of views related by landmarks and distinct visual events embedded in a representation that is more topological than metric. An ALV solely dependent on optical imagery will have to deal with the huge variability in the appearance of objects. Experience has shown that even road surfaces have highly variable visual characteristics. Alternatively, a few pieces of highly pre-selected visual information can serve to verify predictions from a reliable and detailed terrain model and precise position and range sensors.

We call a general object model a schema. A schema can represent perceived, but unrecognized, visual events, as well as recognized objects and their relationships in environmental scenes. The architectural design is focused about the representation, instantiation, and inference over schemas developed by the ALV as it moves through the environment. Schemas are related to similar concepts found in [Hanson et.al. - 78] and [Ohta - 80]. The short term terrain representation consists of schema instantiations that represent accumulated perceptual evidence for objects as attributes and relations that are hypothesized with varying levels of certainty.

Object models are used to organize perceptual processing by integrating descriptive representations with recognition and segmentation control. One aspect of this is the use of different types of attributes and inheritance relations between generic schemas for representation in IS-A and PART-OF hierarchies. A particular object attribute relates three dimensional world properties of an object and sensor dependent view information, either by a set of generic views or viewing procedures. These viewing attributes are also inherited and modified according to different object types. In many systems, objects are treated as lists of attributes that are matched against extracted image features. Here they are treated as specifying an active control process that directs image segmentation by specifying grouping procedures to extract and organize image structures.

Another critical aspect of the architecture is the various types of spatial, localization relations that deal with uncertainty and learning by associating different types of perceptually derived information with terrain models. For example, local (multi-sensor) viewframes affix sets of schemas and un-recognized perceptual structures into local "robot's-eye" views of an ALV's environment. Path-affixments between local viewframes support fusion of information in time without necessarily corresponding to locations in an a priori grid.

This effort has developed an architecture for terrain and object recognition compatible with the wide range of potential sensor configurations and the different qualities of a priori data.

There has been work in artificial intelligence, computer vision, and graphics that satisfy the individual requirements for object modeling capabilities, but little has been done to integrate them. To date, there is no vision system that can interpret general natural scenes, although some can deal with restricted environments [Hanson et.al. - 78] while other systems are restricted to artificial objects and environments. Brooks' [Brooks - 84] representation based on generalized cylinders meets, or could be extended to deal with, many of these functions. It has well defined shape attribute inheritance between a set of progressively more complex object models, and affixment relations that could be generalized to handle uncertainty. It can also be used to generate constraints on image features from object models. Nonetheless, the system built around this representation has had limited success beyond dealing with essentially orthographic views of geometrically well defined man-made objects. This appears to be partially because the constraints on image structures generated from the abstract instances of object models are too general to generate initial correspondences between models and image structures. Brook's system also used an impoverished set of image descriptions, and the object models could not direct the segmentation process directly during their instantiation. The majority of work in terrain modeling deals with how well a representation can realistically model three dimensional terrain, but not how it is used for recognition. The simplicity of a model that is described by a few parameters is not useful for recognition unless it can direct constrained searches against image data. For example, Pentland's [Pentland - 83] use of fractals satisfies aspects of descriptive adequacy for natural terrain, but has been less effective for recognition. Kuipers [Kuipers - 82] has produced an interesting terrain model for learning and handling uncertainty, but it is non visual. Related to this is Kuan's [Kuan - 84] object based terrain representation for planning that is organized in terms of distinct, modifiable objects, but is also not associated with sensor derived processing results.

2. ARCHITECTURE OVERVIEW

The system architecture consists of several databases and inference processes. The inference processes transform the databases, creating additional data structures, and modifying the existing ones. The task interface focuses attention in system processing and monitors progress toward system task goals. This high level architecture is depicted in Figure 2-1. The boxes with square corners in this figure represent databases, the ellipses represent inference processes, and arrows indicate dataflow.

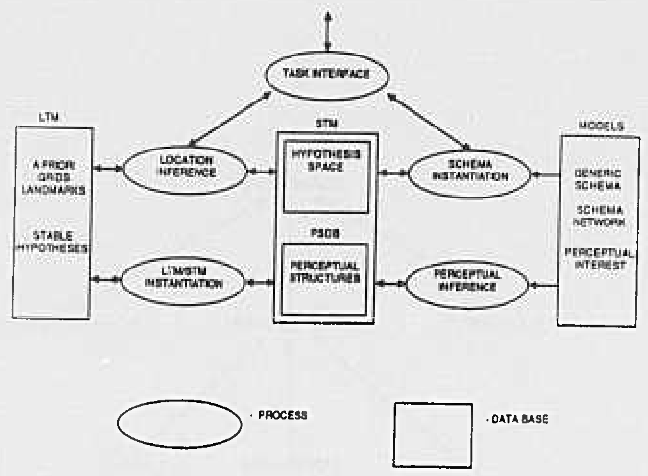


Figure 2-1: Terrain Modeling and Recognition System Architecture

2.1 SYSTEM DATABASES

At the highest level there are three databases. These are the short term memor (STM), long term memory (LTM), and generic models.

The STM acts as a dynamic scratchpad for the vision system. It has two sub-areas, a perceptual structures database (PSDB) and a hypothesis space. The PSDB includes incoming imagery from sensors, immediate results of extracting image structures such as curves, regions and surfaces, spatial/temporal groupings of these structures, and results of inferring 3D information.

The hypotheses space contains statements about objects and terrain in the world. A hypothesis is represented as an instantiated schema. The schema points to the various perceptual structures in the PSDB that provide evidence that the object represented by the schema (such as a terrain patch, road, tree, etc.) exists in the world. Other types of hypotheses include grids, viewframes, and viewpaths. Grids are a special type of terrain representation that contain elevation information and are derived from range data or successive depth maps from motion stereo. Viewpaths, as partially ordered sequences of viewframes, give space/time relationships between hypotheses. Viewframes are sets of hypotheses that correspond to what can be seen from a localized position. A hypothesis with no associated perceptual structures is a prediction. As structures and localization are incrementally added to a hypothesis, it progresses on the continuum from predicted to recognized. Hypotheses that have enough evidence associated with them to be considered recognized and stable, are moved to the LTM.

The LTM stores a priori terrain representations, the long term terrain database, and hypotheses with enough associated evidence to be considered visually stable. A priori data concerning elevation and terrain type information, as well as knowledge of specific landmarks are stored in the LTM. A viewframe, representing a certain location in the world is stored in the LTM if the evidence associated with it could be re-used to recognize the local environment if it was re-encountered. Consistency of one hypothesis with another is not required for storage in the LTM.

The model space stores generic object models, the inheritance relations of the (model) schema network, and a set of image structure grouping processes and rules for evaluating image structure interestingness. Generic models are used dynamically to instantiate and guide search processes to associate evidence to an object instance. Inheritance relations are used by various schema inference procedures to propagate structures, attributes and relations between object instantiations. For instance, the generic two-lane-road schema has an "IS-A" relationship to the generic road schema. It follows, based on the inheritance models, that an instantiation of the two-lane-road schema will inherit the more general characteristics of the generic road schema that in turn inherits the more general characteristics of a terrain patch. Unlike the STM and LTM, the model space is not modified by inference processes.

2.2 INFERENCE PROCESSES

At the highest level, there are five different sorts of inference processes in the vision system. These are perceptual inference, location inference, object instantiation, LTM/STM instantiation, and the task interface.

The PSDB is initialized with the output of standard multi-resolution image processing operations for smoothing, edge extraction, flow field determination, etc. Much subtler inference is required for grouping processes that produce connected curves, textures, surfaces, and temporal matches between image structures. These grouping operations are typically model guided. There are generic models (which may be task dependent) of what constitutes "interestingness" of an image structure.

The hypothesis inference processes produce tasks for the perceptual processes. These may be satisfied by simple queries over the PSDB such as "find all long lines in this region of image", where "long", "line" and "region" are suitably interpreted. Queries can be more complex, requiring, for instance, temporal stability, such as "find all homogeneous green texture regions that are matched (i.e., remain in the field of view) over at least two seconds of imagery", where, again, qualitative descriptors are rigorously defined. Alternatively, the requested perceptual structures may be dynamically extracted. In this case, a history of the processing attempts and results are maintained. If similar requests are made later, such as if we were to view the same environment from a different perspective, these processing histories could be used to recall a processing sequence that produced successful results.

Location processes include a number of different modes of spatial location representation and inference. While exact location information is used when it is available, a key concept is the qualitative representation of relative location. This is fundamental, because the problem of acquiring terrain knowledge from moving sensors involves handling perceptual information that arises from multiple coordinate systems that are transforming

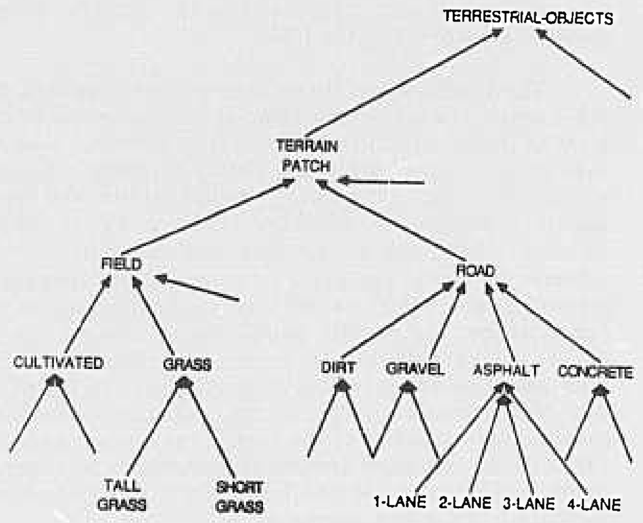


Figure 2-2: IS-A Hierarchy-1

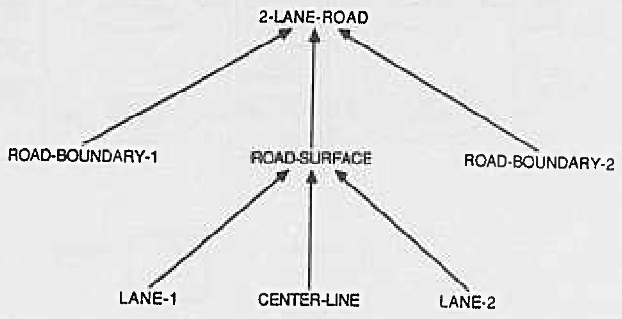


Figure 2-3: Part of Hierarchy-1

in time. The basic approach to location inference is to represent the location of world objects in a qualitative manner that does not require the full knowledge of continuous transformations of sensor coordinates relative to the vehicle the sensors are mounted on, or of transformations of vehicle coordinates relative to the terrain.

The main structures involved in location inference are viewframes, viewpaths, and grids. Viewframes represent both metric location information about world objects derived from range sensors and view-based location information about the directions in which objects are found derived from passive sensor data.

Generic schemas are models of world objects that include information and procedures on how to predict and match the object models in the available sensor data. Besides representing 3D geometric constraints, 2D-3D sensor view appearance including effects of change in resolution and environmental effects such as season, weather, etc., schemas also indicate contextual relationships with other objects, type and spatial constraints, similarity and conflict relations, spatial localization, and appearance in viewframes.

Object schema instantiation may occur by model-driven prediction from a priori knowledge, or directly from another instantiation and a PART-OF relation. The other instantiation process may also occur by matching a distinctive perceptual structure to a schema appearance instance. This sort of "triggering" is more common in situations where there is little a priori information to guide prediction. Object instantiations generate queries to the PSDB grouping process in order to complete matching.

A key idea in object instantiation processing is inference over the model schema network hierarchies. Direct representation and inference over a large enough body of world objects to accomplish outdoor terrain understanding requires very large memory and proportionately lengthy inference procedures over that memory space. Hierarchical representation makes a significant reduction in storage requirements; furthermore, it lends itself naturally to matching schema to world objects at multiple levels of abstraction, thus speeding the inference process. Two basic hierarchies are the IS-A and PART-OF trees.

IS-A hierarchies represent the refinement of object classification. Figure 2-2 shows part of an IS-A hierarchy for terrain representation. The level of terrestrial-object tells us that we will not see evidence of any schema instance below this node as perceptual structures surrounded by sky. At the level of terrain-patch we pick up the geometric knowledge of adherence to the ground plane, while information stored at the level of a road schema constrains the boundaries of a terrain patch to be locally linear (with other constraints). Types beneath road add critical appearance constraints in color and texture, while the final refinement level in the IS-A hierarchy, the number of lanes, further constrains size parameters inherited from the road schema.

PART-OF hierarchies represent the decomposition of world objects into components, each of which is, itself, another world object. Figure 2-3 shows a PART-OF hierarchy decomposition for a generic 2-lane-road. PART-OF hierarchies contain relative geometric information that is useful in prediction and search.

As object instantiation inference reasons up and down schema network hierarchies, incrementally matching perceptual structures and other data to instances of object appearance in the world, a history mechanism records the inference processing steps, parameters and results. This dynamic data structure is called the schema instantiation structure. One important aspect of this structure is that it can used to extract the inference and processing sequence(s) that worked earlier to see the same object, or ones that are similar. This accounts for the fact that distinctiveness in image appearance is an idiosyncratic process that depends upon many factors which are difficult to model and control, such as current motion, wind, varying outdoor illumination, etc.

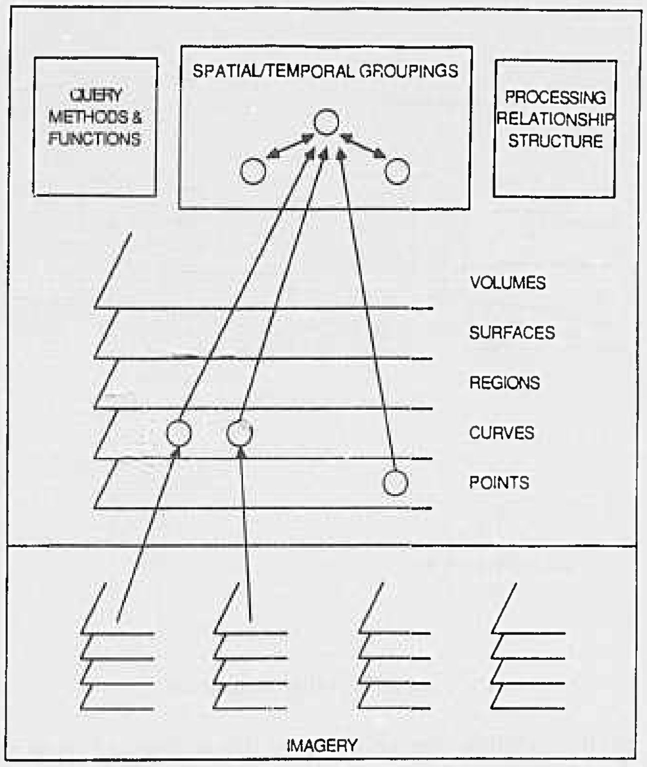

QUERY METHODS & FUNCTIONS
SPATIAL/TEMPORAL GROUPINGS
PROCESSING RELATIONSHIP STRUCTURE
VOLUMES
SURFACES
REGIONS
CURVES
POINTS
IMAGERY

Figure 3-1: Perceptual Structure Data Base (PSDB)

3. PERCEPTUAL PROCESSING AND THE PSDB

Perceptual processing is concerned with organizing images into meaningful chunks. The definition of "meaningful" and the development of explicit criteria to evaluate segmentation techniques involves, from a data-driven perspective, that the chunks have characterizing properties, such as regularity, connectedness, and not tending to fragment the image. From a model-driven point of view, segmentation appropriateness corresponds to the extent to which the pieces can be matched to structures and predictions derived from object models. From either perspective, a basic requirement is that image segmentation procedures find significant image structures, independent of world semantics, in order to initialize and cue model matching. This allows for the extraction of world events such as surfaces, boundaries, and interesting patterns independent of understanding perceptions in the context of a particular object. These, in turn, are useful abstractions from image information to match against object models or describe the characteristics of novel objects.

The Perceptual Structure Data Base (PSDB), conceptualized in Figure 3-1, contains several different types of information. These are classified as images, perceptual objects, and groups. Images are the arrays of numbers obtained from the different sensors and the results of low level image processing (such as contour extraction and region growing routines) that produce such arrays. It is difficult for the symbolic/relational

```
Curve: #<CURVE 175505263>
Point1: (205 291)
Point2: (274 205)
Length: 01
Grid: 6
Points: (205 291) (206 292) (207 293) ...... (276 207) (275 206) (274 205)
AVG-INTENSITY: 159.27315
SUCCESSIVE-GRADIENT-DIFFERENCE: 1.7660371 1.4702858 1.0026073 ......
         0.5625056 0.6442404 0.7240415
GRADIENT: (-2.0555900 3.1515503) (-4.0343933 1.8365173) (-5.415619 1.3096924) ......
         (-0.03042163 -2.917309) (0.56552124 -2.6931152) (1.0051135 -2.1800733)
CONTRAST: 5.7423205
AVG-GRADIENT: -5.7336416 0.31559697
LOC: (#<CURVE 175505520>
       (#<CURVE 175327156>)
       (#<CURVE 175505525>
           (#<CURVE 175327163>)
           (#<CURVE 175505532>
            (#<CURVE 175327170>)
            (#<CURVE 175327175>))))
```

Figure 3-2: Curve Example

representations used for object models, such as schemas, and the processing rules in computer vision systems, to work directly with an array of numbers. Therefore, there are many spatially-tagged, symbolic representations used in image understanding systems that describe extracted image structures such as the primal sketch [Marr - 82], the RSV structure of the VISIONS system [Hanson et.al. - 78], and the patchery data structure of Ohta [Ohta - 80]. We found it useful to build such a representation around a set of basic perceptual objects corresponding to points, curves, regions, surfaces, and volumes.

Groupings are recursively defined to be a related set of such objects. The relation may be exactly determined, as in representing which edges are directly adjacent to a region, or they may require a grouping procedure to determine the set of objects that satisfy the relationship. Groupings can occur over space, e.g., linking texture elements under some shape criteria such as compactness and density, or over time, as in associating instances of perceptual structures in successive images. We stretch the concept a bit, so that groupings also refer to general non-image registered perceptual information, such as histograms.

3.1 INITIALIZATION OF THE PSDB

Whenever new sensor data is obtained, a default set of operations are performed to initialize the PSDB. Edges are extracted at multiple spatial frequencies and decomposed into linear subsegments. The edges are extracted into distinct connected curves, and general attributes such as average intensity, contrast, and variance are associated with them. Similar processing is performed for regions extractions. Histograms are computed with respect to a wide range of object based and image based characteristics in a pyramid like structure. These default operations are used to initialize bottom-up grouping processes and schema instantiations. These, in turn, determine significant structures using heuristic interestingness rules to prioritize the structures for the application of grouping processes or object instantiations.

3.2 IMAGES

Images are the data arrays derived from the optical and laser range sensors and the results of image processing routines for operations including histogram-based segmentation, different edge operators, optic flow field computations, and so forth. Associated with images are several attributes for time of acquisition, relevant sensor parameters, etc. Processing history is maintained in the processing relationship structure that keeps track of the processing history of all objects in the PSDB.

3.3 PERCEPTUAL OBJECTS

Points, curves, regions, surfaces, and volumes are basic types of perceptual structures that are accessible to object instantiations and grouping processes. An example instance of a curve structure is shown in Figure 3-2. This figure shows many common representational characteristics of perceptual objects. There are default attributes associated with particular objects, such as end-points, length and positions for a curve. There is also an associated attribute-list mechanism for incorporating more general properties with an object. This list is accessible by keywords and a general query mechanism using methods specific to the particular associated attribute. The associated attributes in the example are shown in capital letters. There are many types of attributes that can be consistently associated with a curve using this mechanism.

A useful representation for performing geometric operations and queries over objects is the OBJECT LABEL-GRID (or GRID: in the example curve. The number 6 indicates the index of this structure). This is an image where each pixel contains a vector of pointers back to the set of perceptual objects and groups which occupy that position. This allows geometric operations to be performed directly on the grid. Filtering operations can be applied to the OBJECT LABEL GRID to restrict processing based upon attributes associated with objects. Various types of masks can be associated with objects to reflect a directional or uniform neighborhood to determine object relationships in the OBJECT LABEL GRID.

3.4 GROUPS

A group is a set of related perceptual objects. The relation can be determined directly by a query over an object and those surrounding it, as in finding the set of curves within some distance of a given region. Alternatively, it may require a search process to find the set of objects meeting some, potentially complex, criteria. For example, an ordered set of curves can be grouped together using thresholds on allowable changes in the average contrast and orientation of successive elements. By expressing the grouping process as a search over a state space of potential groups, each group becomes a potential hypothesis in the PSDB. Groups can also reflect temporal relationships: this occurs in matching structures in successive images. A relational grouping procedure is shown in Figure 3-3 for the determination of nearby parallel lines with opposite contrast directions. This is done for a linear segment by first extracting

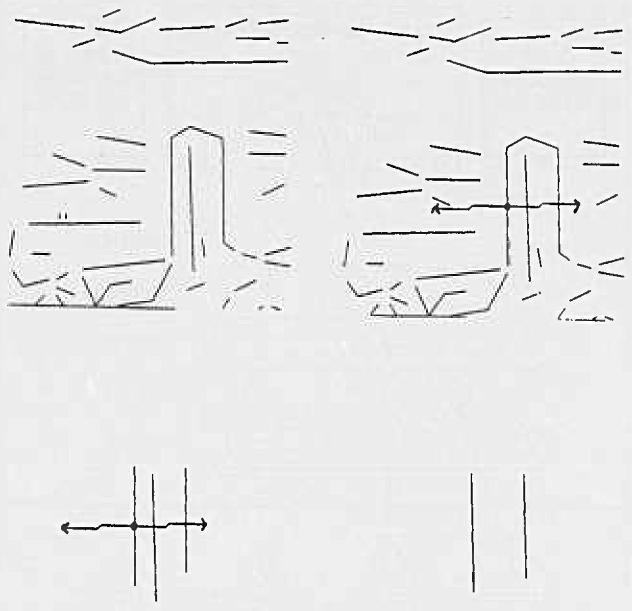


Figure 3-3: Parallel Grouping

nearby neighbors using a narrow mask oriented perpendicular from the segment at its mid-point. The intersection of this mask with points in the label grid are determined, and then each candidate is evaluated by checking if it is within allowable thresholds for length, contrast, and orientation. It is then ordered with respect to the smallest magnitude of the difference vector computed from the average gradients. The grouping processes can either produce the best candidate as a potential grouping, or some set of them.

Two different types of grouping processes have been developed: measure-based and interestingness-based. The measure based grouper is a generalization of established edge and region linkers [Martelli - 76]. It uses a measure consisting of:

1) some value to be optimized, such as length, minimal curvature, compactness, or a composite scalar value

2) local constraints on allowable changes in attributes

3) global thresholds on attributes

The measure and associated constraints are optimized by a best first search returning several ordered candidate groups. The measure to be used can be associated with a prediction from an object model for substance or shape characteristics. The measure to be optimized can also be determined directly from initially extracted objects by selecting those that are extreme in some attribute or are correlated with the attributes of surrounding objects to derive a measure to be optimized.

The measure based grouper is currently being generalized into one based on interestingness. It involves the basic processing loop shown in Figure 3-4. Initially, basic perceptual objects including curves, regions, junctions and their associated attributes are extracted using conventional techniques. Extracted objects are represented in label grids to express spatial neighborhood operations over the objects. A uniform neighborhood is established for each object, and directed relations are formed with the adjacent objects in each neighborhood. These relations are represented in a small number of types of match relationships that contain descriptions of the correlation of attributes, subcomponent matching, and composite properties.

Selected attributes of the extracted perceptual objects and the match structures are then sorted into lists with pointers back to the associated objects. These lists are for attributes such as size, average feature values, variance of feature values, compactness, the extent of correlation between the components and attributes of different structures, and the number of groups an object is involved in. These different rankings are then combined using a selection criteria to choose the set of interesting perceptual objects and relationships. The selection criteria sets the required position in different subsets of the sorted attribute lists. An example is to find 100 largest objects in the top 10 of any of the attribute correlation lists. The selection criteria is modifiable during processing and is meant to reflect the influence of model-based predictions.

Interestingness is used to focus the application of grouping rules to a selected set of objects and relations between objects indicated in match structures. The grouping rules then combine perceptual objects to form new perceptual objects, or groups, based upon the type of relation between the objects. Neighborhoods are established with respect to these derived groups to form new relationships. These in turn are sorted in the attribute lists with respect to the previously extracted perceptual objects. In addition to the relations established in uniform neighborhoods, for some groups, non-uniform relations are also established. Processing can continue indefinitely as less and less interesting relations become candidates for the application of grouping rules. Explicit criteria are needed to stop processing; e.g., we can limit processing time, determine when there is a uniform covering of the image with extracted groups, or when structures belong to unique groups.

These operations are performed by virtual processors called grouping nodes. Grouping nodes are seen as covering regular and adjacent portions of an image area (not necessarily of a single image, because there can be multiple images in a motion sequence). The image area contains some portion of a label plane for accessing the objects based upon their spatial dispositions as well as object-based associated attributes. The grouping nodes are further organized in a hierarchical pyramid shown in Figure 3-5. Each node is connected to its adjacent neighbors and has a parent and descendants. The transfer of information between nodes at different levels is based upon interestingness. Lower level processes send their most interesting structures up the hierarchy. There

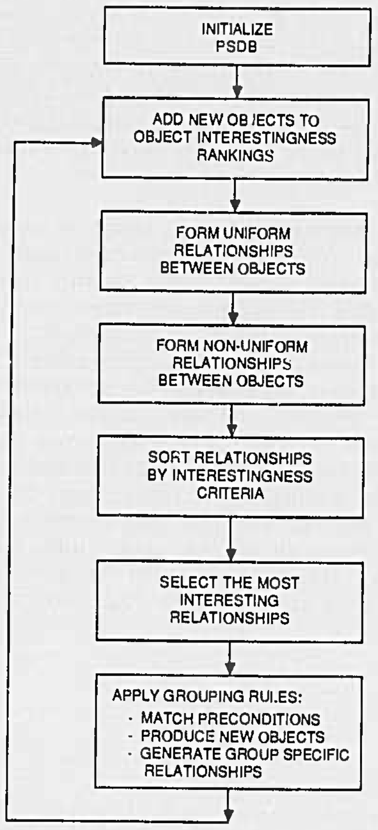

INITIALIZE PSDB
ADD NEW OBJECTS TO OBJECT INTERESTINGNESS RANKINGS
FORM UNIFORM RELATIONSHIPS BETWEEN OBJECTS
FORM NON-UNIFORM RELATIONSHIPS BETWEEN OBJECTS
SORT RELATIONSHIPS BY INTERESTINGNESS CRITERIA
SELECT THE MOST INTERESTING RELATIONSHIPS
APPLY GROUPING RULES:
- MATCH PRECONDITIONS
- PRODUCE NEW OBJECTS
- GENERATE GROUP SPECIFIC RELATIONSHIPS

Figure 3-4: Grouping Processing Flow

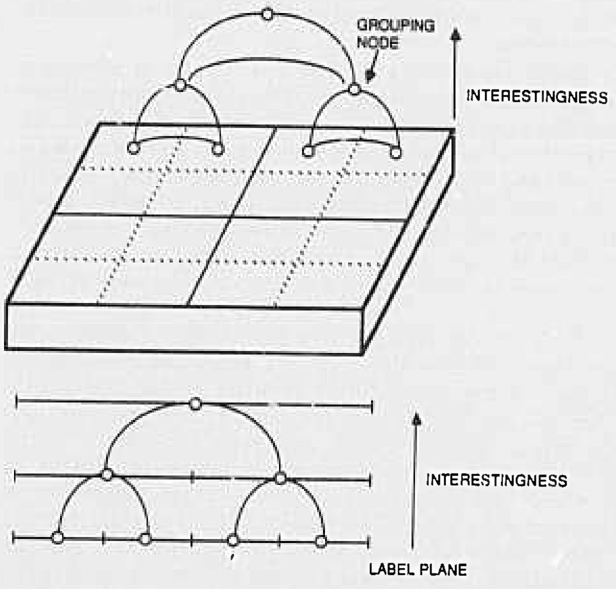


Figure 3-5: Grouping Architecture

are several effects of this. One is that it allows a uniform processing to occur at different levels, so grouping rules can be applied to objects at different levels of interestingness. It also allows relations between nonspatially adjacent structures to be handled in a uniform architecture. It also partitions perceptual structures in a way that corresponds to different levels of control in instantiation of object models.

Organizing segmentation in terms of grouping processes has many advantages for a model based vision system. The grouping processes can be run automatically from extracted significant structures based upon perceptually significant, though non-semantic criteria. Thus, connected curves of slowly changing orientation or compact, homogeneous regions can be extracted purely on perceptual criteria. These image structures correspond to world structure and events, and they are useful for initializing schema instantiations. They correspond to the qualitative image predictions associated with more general schemas. An inference process for compilation from an object model into grouping processes, allows model based vision to have a very active character quite different from single-level attribute matching.

4. SCHEMAS

Schemas represent hypotheses about objects in the world. The process of schema instantiation creates an instance of a schema together with evidence for that schema. Evidence consists of structures in the PSDB, a priori knowledge stored in the LTM, predictions derived from location inference, and relations to already instantiated schema.

Table 4-1 shows the various slots and relationships in a generic schema. Although this data structure has a frame-like appearance, it is useful to view the schema as a semantic net structure, with slots representing nodes in the net and relationships representing arcs. Schema instantiation inference reasons from a (partially) instantiated node, follows arcs, and infers procedures to execute from the sum of its acquired information in order to obtain more evidence to further instantiate the schema.

The schema network is a generic set of data structures that indicate the a priori relationships between schemas. A key part of this network is the inheritance hierarchies that indicate which descriptions and relationships can be inherited from schema to schema. Inheritance hierarchies allow efficient matching of objects in the world against sensor evidence from progressively coarser to finer levels. As reasoning moves from coarser to finer levels of description in model-based schema instantiations, the schemas inherit descriptive bounds and add new descriptions, and also add constraints to inherited ones. For example, the system may first recognize an object as a terrain patch (because it lies on the ground plane). A road is a type of terrain patch (see Figure 2-1, that adds linear boundary description, and constrains the visual image appearance of the terrain patch schema in the color and texture descriptors. The two basic types of schema network inheritance hierarchies are IS-A and PART-OF.

- SCHEMA TYPE
- SCHEMA NAME
- SCHEMA INSTANTIATION STRUCTURE
- 3D DESCRIPTION
  - o SHAPE
  - o SIZE
  - o COLOR
  - o TEXTURE
  - o INDEX TO SENSOR VIEWS
- SENSOR VIEWS
  (FOR EACH SENSOR:)
  (FOR EACH VIEW:)
  - o PROJECTION RELATIONS
    - - PROJECTION FUNCTION
    - - 3D BACK CONSTRAINTS
  - o DISTINCTIVE IMAGE BASED EVIDENCE
  - o PERCEPTUAL STRUCTURE
- COMPONENTS
  - o MUST HAVE
  - o MAY HAVE
  - o 3D SPATIAL RELATIONSHIPS
  - o VIEW DEPENDENT RELATIONSHIPS
- PART OFS
- CLASSIFICATIONS
  (POINTS UP THE IS-A HIERARCHY ONE LEVEL)
- CONTEXTUAL RELATIONSHIPS
  - o ALWAYS OCCURS WITH
  - o SOMETIMES OCCURS WITH
  - o NEVER OCCURS WITH
  - o CONFUSED WITH
  - o SIMILAR TO
- LOCATIONAL INFORMATION
  - o LOCAL MULTI-SENSOR FRAME AFFIXMENTS
  - o GRID AFFIXMENTS
  - o 3D SPATIAL RELATIONSHIPS WITH
- RECOGNITION STRATEGIES

Table 4-1: Generic Schema Data Structure

Below is a brief explanation of each of the slots and relationships in the generic schema data structure. Schema type refers to the generic name of the schema in the IS-A hierarchy. Schema name is the identification of the schema instance, e.g., if the schema type is "road" then the schema name might be "highway 101". The schema instantiation structure maintains the control history of the schema recognition inference processes for this schema.

The 3D description is an object-centered view of the world object represented by the schema. It includes its 3D geometry and shape description, actual size, and inherent color and texture (as opposed to how its color and texture might appear to a particular sensor). Note that this is the description that matches the schema-object before looking at its structure refined into components. For example, the 3D geometric description of a tree schema does not separate the canopy from the trunk, but gives a single enclosing volume as its representation. The volumetric descriptions of the trunk and canopy appear as the 3D descriptors on their schema further down the PART-OF hierarchy. Thus, inferring down the PART-OF hierarchy corresponds to increasing the resolution of the view of the object represented by the schemas.

The sensor views are descriptions of the stable or frequently occurring appearances of the schema object in imagery. This description is intended to be used for image appearance prediction, evidence accrual for instance recognition, 3D shape inference, and location inference. The reason for storing or runtime generation of explicit (parametrized) image views is that the perceptual evidence matches to these descriptions, not to the three dimensional ones.

The distinctive image appearance slot holds descriptions of perceptual structures that are likely to occur bottom-up in the PSDB. They provide coar triggers for instantiating the schema object hypothesis without prediction.

The perceptual structure is the dynamically created PSDB query history generated by the schema instantiation as it attempts to fill in evidence matching the various schema slots and relations. The instantiator can re-use successful branches of perceptual structures to improve its recognition speed as it continues to view other instances of the same generic schema type.

Components are pointers to other schema that represent sub-parts of the schema object. They are finer resolution description of the schema, one level down on the PART-OF hierarchy. The MUST-HAVE components are assumed to be parts the represented object must have to exist, although the schema may be instantiated without observing them all. Occasionally occurring components, such as center-lines on roads, can be stored in the MAY-HAVE slot. Spatial relationships between components as they make up the schema object are listed at this level also. Relationships can also be stored on a view dependent basis. These relationships access the sensor-view dependent data in that slot. PART-OF's point upward one level on the PART-OF hierarchy, indicating that this schema is a component of another schema.

Classification points upward and downward one level on the IS-A hierarchy. There may be more than one such pointer, which is to say that the IS-A hierarchy may be partially ordered.

Contextual relationships indicate spatial/temporal consonance or disconsonance between groups of schema types, omitting those which are already indicated in the PART-OF and IS-A hierarchies. Schema that ALWAYS or never-occur with the given one can be used strongly for belief or dis-belief in the schema instance and as focus of attention mechanisms within the instantiation process. SOMETIMES occurs with relationships that are used to store the spatial-temporal aspects of schemas relative appearance in the viewed environment.

CONFUSED-WITH and SIMILAR-TO relationships indicate schema that may be mistaken for the given one, but for different reasons. One schema may be confused with another because they share common evidence pieces, but for which there are sufficient descriptors to disambiguate. Two schema are similar if there is sufficient ambiguity in their appearances, and therefore the available perceptual evidence, that they may be indistinguishable without contextual reasoning. For example, tall grass may be confused with wheat from coarse shape and texture evidence, but can often be disambiguated by color descriptors or finer resolution examination of structure (because of wheat berries, for example). However, roads are similar to runways because they cannot necessarily be distinguished by their intrinsic appearance, no matter how detailed or accurate the descriptors and evidence. Contextual reasoning, e.g., the presence of aircraft on the runway, global curvature of the road, etc. is required.

Locational information points at the various viewframes the schema appears in and inferred 3D relationships with other world objects.

Recognition strategies are prioritization cues for the schema instantiation processes that suggest inference chains likely to pay off to match this schema instance against sensor evidence.

The recognition strategies slot in the schema data structure prioritizes inference approaches relevant to this schema. These approaches include search for components, search for part of schema instance, search on weaker classification, relations with other schema instances, and PSDB matching.

Search for COMPONENTS and search for PART-OF are both inferences along the PART-OF hierarchy in different directions. The instantiator searches the relevant slot to see if there are components to search for or another object of which this schema is a component. If the COMPONENT or PART-OF schemas exist, they can be accessed to continue the inference. Otherwise, each causes an instantiation of the missing schema to be generated as a prediction. Instantiation control can be transferred at this point to the COMPONENT or PART-OF schema. The schema inference process maintains its thread of reasoning relevant to the schema in the schema instantiation structure slot.

5. LONG TERM TERRAIN DATABASE

The long term terrain database is part of LTM. It stores the data necessary for a mobile robot to perform vision-based navigation and guidance, predict visual events, such as landmarks and horizon lines, and to update and refine maps.

The long term terrain database contains a priori map data including government terrain grids, elevation data, and schemas representing instances of stable visual events recorded while traversing paths in the environment. The use of a priori map and grid data to predict percepts and to help guide image segmentation is shown in Section 6. The following presents a summary of a structure for spatial representation and inference that enables a robot to navigate and guide itself through the environment.

We first define the notion of a geographic "place" in terms of data about visible landmarks. A place, as a point on the surface of the ground, is defined by the landmarks and spatial relationships between landmarks that can be observed from a fixed location. More generally, a place can be defined as a region in space, in which a fixed set of landmarks can be observed from anywhere in the region, and relationships between them do not change in some appropriate qualitative sense. Data about places is stored in structures called viewframes, boundaries and orientation regions.

Viewframes provide a definition of place in terms of relative angles and angular error between landmarks, and very coarse estimates of the absolute range of the landmarks from our point of observation. Viewframes allow the system to localize its position in space relative to observable local landmark coordinate systems. In performing a viewframe localization, observed or inferred data about the approximate range to landmarks can be used. Errors in ranging and relative angular separation between landmarks are smoothly accounted for. A priori map data can also be incorporated. A viewframe is pictured in Figure 6-11.

A viewframe encodes the observable landmark information in a stationary panorama. That is, we assume that the sensor platform is stationary long enough for the sensor to pan up to 360 degrees, to tilt up to 90 degrees (or to use an omni-directional sensor [Cao et.al. - 86]), to recognize landmarks in its field of view, or to buffer imagery and recognize landmarks while in motion.

A sensor-centered spherical coordinate system is established. It fixes an orientation in azimuth and elevation, and takes the direction opposite the current heading as the zero degree axis. Then two landmarks in front of the vehicle, relative to the heading, will have an azimuth separation of less than 180 degrees. If we assume that no two distinguished landmark points have the same elevation coordinates (i.e., no two distinguished points appear one directly above the other) then a well-ordering of the landmarks in the azimuth direction can be generated. We can speak of the landmarks as being "ordered from left to right". The relative solid angle between two distinguished landmark points is now well defined.

Under the above assumptions, the system can pan from left to right, recognizing landmarks, L_i, and storing the solid angles between landmarks in order, denoting the angle between the i-th and j-th landmarks by Ang_{ij}. The basic viewframe data are these two ordered lists, $(L_1, L_2, \ldots)$ and $(\mathrm{Ang}_{12}, \mathrm{Ang}_{23}, \ldots)$. The relative angular displacement between any two landmarks can be computed from this basic list. In [Levitt et.al. - 87] we show how to use this data to essentially parametrize all possible triangulations of our location relative to a set of simultaneously visible landmarks. This localizes the robot's position in space relative to a local landmark coordinate system.

Viewframes contain two basic dimensions of data: the relative angles between landmarks, and the estimated range (intervals) to the landmarks. If we drop the range information, we are left with purely topological data. That is, it is impossible, using only the relative angles between landmarks, and no range, map or other metric data, to determine the relative angles between triples of landmarks, or to construct parametric representations of our location with respect to the landmarks. Nonetheless, there is topological localization information present in the ordinal sequence of landmarks; there is a sense in which we can compute differences between geographic regions, and observe which region we are in.

The basic concept is to note that if we draw a line between two (point) landmarks, and project that line onto the (possibly not flat) surface of the ground, then this line divides the earth into two distinct regions. If we can observe the landmarks, we can observe which side of this line we are on. The "virtual boundary" created by associating two observable landmarks together thus divides space over the region in which both landmarks are visible. We call these landmark-pair-boundaries (LPB's), and denote the LPB constructed from the landmarks L_1 and L_2 by LPB(L_1,L_2).

Roughly speaking, if we observe that landmark L_1 is on our left hand, and landmark L_2 is on our right, and the angle from L_1 to L_2 (left to right) is less than 180 degrees, then we denote this side of, or equivalently, this orientation of, the LPB by $[L_1\ L_2]$. If we stand on the other side of the boundary, LPB(L_1,L_2), "facing" the boundary, then L_2 will be on our left hand and L_1 on our right and the angle between them less than 180 degrees, and we can denote this orientation or side as $[L_2\ L_1]$ (left to right).

More rigorously, define:

orientation-of-LPB(L_1,L_2)

$$= \mathrm{sign}(\pi - \Theta_{12}) = \begin{array}{l} +1 \text{ if } \Theta_{12} < \pi \\ 0 \text{ if } \Theta_{12} = \pi \\ -1 \text{ if } \Theta_{12} > \pi \end{array}$$

where Θ_{12} is the relative azimuth angle between L_1 and L_2 measured in an arbitrary sensor-centered coordinate system. Here, an orientation of +1 corresponds to the $[L_1\ L_2]$ side of LPB(L_1,L_2), -1 corresponds to the $[L_2\ L_1]$ side of LPB(L_1,L_2) and 0 corresponds to being on LPB(L_1,L_2). It is a straightforward to show that this definition of LPB orientation does not depend on the choice of sensor-centered coordinate system.

LPB's give rise to a topological division of the ground surface into observable regions of localization, called orientation regions. Crossing boundaries between orientation regions leads to a qualitative sense of path planning based on perceptual information. The three levels of spatial representation given by map or metric data, viewframes and orientation regions are pictured in Figure 5-1. A natural environmental representation based on viewframes recorded while following a path is given by two lists, one list of the ordered sequence of viewframes collected on the path, and another of the set of landmarks observed on the path. We call the viewframe list a viewpath. The landmark list acts as an index into the viewpath, each landmark pointing at the

We first define the notion of a geographic "place" in terms of data about visible landmarks. A place, as a point on the surface of the ground, is defined by the landmarks and spatial relationships between landmarks that can be observed from a fixed location. More generally, a place can be defined as a region in space, in which a fixed set of landmarks can be observed from anywhere in the region, and relationships between them do not change in some appropriate qualitative sense. Data about places is stored in structures called viewframes, boundaries and orientation regions.

Viewframes provide a definition of place in terms of relative angles and angular error between landmarks, and very coarse estimates of the absolute range of the landmarks from our point of observation. Viewframes allow the system to localize its position in space relative to observable local landmark coordinate systems. In performing a viewframe localization, observed or inferred data about the approximate range to landmarks can be used. Errors in ranging and relative angular separation between landmarks are smoothly accounted for. A priori map data can also be incorporated. A viewframe is pictured in Figure 6-11.

A viewframe encodes the observable landmark information in a stationary panorama. That is, we assume that the sensor platform is stationary long enough for the sensor to pan up to 360 degrees, to tilt up to 90 degrees (or to use an omni-directional sensor [Cao et.al. - 86]), to recognize landmarks in its field of view, or to buffer imagery and recognize landmarks while in motion.

A sensor-centered spherical coordinate system is established. It fixes an orientation in azimuth and elevation, and takes the direction opposite the current heading as the zero degree axis. Then two landmarks in front of the vehicle, relative to the heading, will have an azimuth separation of less than 180 degrees. If we assume that no two distinguished landmark points have the same elevation coordinates (i.e., no two distinguished points appear one directly above the other) then a well-ordering of the landmarks in the azimuth direction can be generated. We can speak of the landmarks as being "ordered from left to right". The relative solid angle between two distinguished landmark points is now well defined.

Under the above assumptions, the system can pan from left to right, recognizing landmarks, L_i, and storing the solid angles between landmarks in order, denoting the angle between the i-th and j-th landmarks by Ang_{ij}. The basic viewframe data are these two ordered lists, $(L_1, L_2, \ldots)$ and $(\mathrm{Ang}_{12}, \mathrm{Ang}_{23}, \ldots)$. The relative angular displacement between any two landmarks can be computed from this basic list. In [Levitt et.al. - 87] we show how to use this data to essentially parametrize all possible triangulations of our location relative to a set of simultaneously visible landmarks. This localizes the robot's position in space relative to a local landmark coordinate system.

Viewframes contain two basic dimensions of data: the relative angles between landmarks, and the estimated range (intervals) to the landmarks. If we drop the range information, we are left with purely topological data. That is, it is impossible, using only the relative angles between landmarks, and no range, map or other metric data, to determine the relative angles between triples of landmarks, or to construct parametric representations of our location with respect to the landmarks. Nonetheless, there is topological localization information present in the ordinal sequence of landmarks; there is a sense in which we can compute differences between geographic regions, and observe which region we are in.

The basic concept is to note that if we draw a line between two (point) landmarks, and project that line onto the (possibly not flat) surface of the ground, then this line divides the earth into two distinct regions. If we can observe the landmarks, we can observe which side of this line we are on. The "virtual boundary" created by associating two observable landmarks together thus divides space over the region in which both landmarks are visible. We call these landmark-pair-boundaries (LPB's), and denote the LPB constructed from the landmarks L_1 and L_2 by LPB(L_1,L_2).

Roughly speaking, if we observe that landmark L_1 is on our left hand, and landmark L_2 is on our right, and the angle from L_1 to L_2 (left to right) is less than 180 degrees, then we denote this side of, or equivalently, this orientation of, the LPB by $[L_1 L_2]$. If we stand on the other side of the boundary, LPB(L_1,L_2), "facing" the boundary, then L_2 will be on our left hand and L_1 on our right and the angle between them less than 180 degrees, and we can denote this orientation or side as $[L_2 L_1]$ (left to right).

More rigorously, define:

orientation-of-LPB(L_1,L_2)

$$= \mathrm{sign}(\pi - \Theta_{12}) = \begin{array}{l} +1 \text{ if } \Theta_{12} < \pi \\ 0 \text{ if } \Theta_{12} = \pi \\ -1 \text{ if } \Theta_{12} > \pi \end{array}$$

where Θ_{12} is the relative azimuth angle between L_1 and L_2 measured in an arbitrary sensor-centered coordinate system. Here, an orientation of $+1$ corresponds to the $[L_1 L_2]$ side of LPB(L_1,L_2), -1 corresponds to the $[L_2 L_1]$ side of LPB(L_1,L_2) and 0 corresponds to being on LPB(L_1,L_2). It is a straightforward to show that this definition of LPB orientation does not depend on the choice of sensor-centered coordinate system.

LPB's give rise to a topological division of the ground surface into observable regions of localization, called orientation regions. Crossing boundaries between orientation regions leads to a qualitative sense of path planning based on perceptual information. The three levels of spatial representation given by map or metric data, viewframes and orientation regions are pictured in Figure 5-1. A natural environmental representation based on viewframes recorded while following a path is given by two lists, one list of the ordered sequence of viewframes collected on the path, and another of the set of landmarks observed on the path. We call the viewframe list a viewpath. The landmark list acts as an index into the viewpath, each landmark pointing at the observations of itself in the viewframes. For efficiency, the landmark list can be formed as a database that can be accessed based on spatial and/or visual proximity. Visual proximity can be observed, or computed from an underlying elevation grid and a model of sensor and vision system resolution.

The first occurrence of a landmark points at the instantiated schema or perceptual structure in the vision system database that was used to gather evidence in the landmark recognition process. After that, all recognized re-occurrences of this landmark point back at this initial instance. The same is true for the first occurrences and successful re-recognition of LPB's and viewframes. This mechanism allows multiple visual path representations, built at different times, to be incrementally integrated together as they are acquired by using a common landmark indexing/pointer list.

We use an environmental representation for orientation-region reasoning that is a list of oriented LPB's encountered and crossed in the course of following a path. We call such a list an orientation-path. As with viewpaths, there is an associated landmark list that indexes into the orientation-path.

A dynamically acquirable environmental representation that merges the representations for viewpaths and orientation-paths consists of an ordered list interspersing viewframes, LPB crossings, and appearance and occlusion (or loss of resolution) of landmarks, as well as recording the headings taken in the course of following the path over which the environmental map is being built. Thus, we can integrate the representations required for viewframe and orientation region based reasoning with heading and landmark information to formulate an environmental representation that supports hybrid strategies for navigation and guidance. The representation is formed at runtime and consists of multiple interlocking lists of sequential, time ordered, lists of visual events that include those necessary for the navigation and guidance algorithms presented in [Levitt et.al. - 87].

6. PROCESSING EXAMPLE

The following processing example demonstrates the behavior of some implemented system components. These include the format of predictions from the long term terrain model, the extraction of perceptually significant groupings from the PSDB, how an instantiated schema uses grouping processes and queries over the PSDB, and extracting relevant cues for making viewframe localizations in the long term terrain representation.

Figure 6-1 shows the elevation contours and road network in the a priori terrain data from the Martin Marietta ALV test site in Denver which was supplied by the U.S. Army Engineer Topographic Laboratories (ETL). The vehicle position on the road is indicated by the arrow in the figure. From this, we are able to roughly determine the correspondence between an image taken from the road and the terrain data, (the relevant

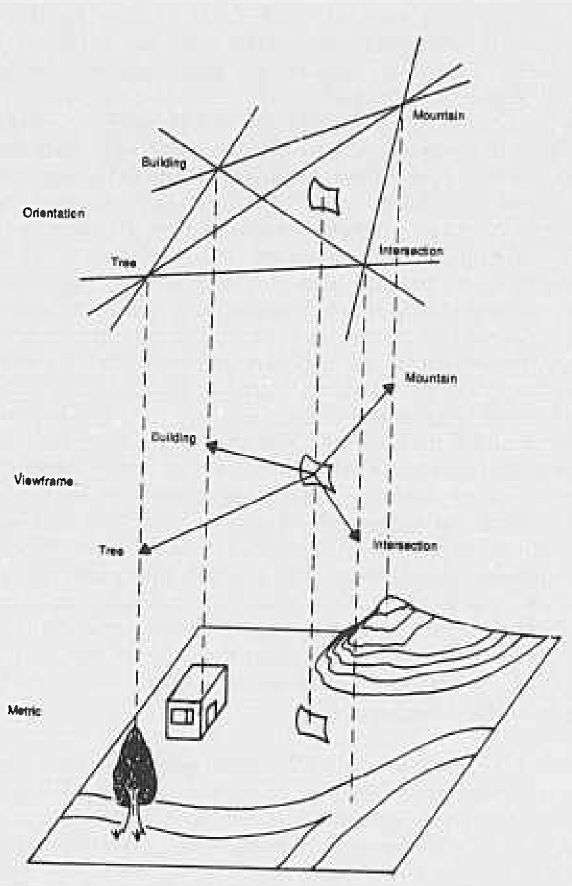


Figure 5-1: Multiple-Levels-of-Spatial Representation

sensor parameters were not available). Figure 6-2 shows the terrain and feature classification supplied with the a priori data. These correspond to sets of image overlays in register with the elevation data. The road network is stored as a set of curve objects that is decomposed into linear segments with supplied attributes, such as road material and width. Terrain patches are extracted as regions from terrain type information and parametric surface fits to the a priori elevation data.

Figure 6-3 shows how the grid registered terrain data is instantiated into STM to form a predicted segmentation. The grid data regions from connected analysis correspond to schema instances in the Long Term terrain memory. Established surface display techniques are used to project the elevation with the associated schema instances to form a predicted view. Image positions are then labeled with their associated schema instances. Additionally, there are many schema instances, ordered by depth, at the corresponding image locations. The resulting predicted segmentation is processed as an abstract image where critical perceptual events are determined by size, adjacencies across occlusion boundaries, or types of terrain with high semantic contrast, such as water, fields, or man-made structures. The perceptual structures are merged together based upon distances and semantic type to yield predictions at different resolutions.

Figure 6-4 shows the predicted terrain patches for the vehicle positioned with respect to the terrain in Figure 6-1. Figure 6-5 shows the predicted segmentation after filtering to pull out the horizon line and road/terrain discontinuities for roads near the vehicle. This data is quite coarse (30m sampling), and image areas in the foreground are highly composite containing instances of road and the adjacent grassy fields. Nonetheless, the predicted segmentation yields a qualitative description of predicted image features that is sufficient to initialize and direct grouping processes to find corresponding image features and relationships. The key characteristics of the predicted segmentation are that the vehicle is on a flat plane, and that its field of view consists of road and grassy field terrain patches with some mountains in the distance. Predictions of the dirt road off to the right and the intersection are made from the road-network and the elevation information stored along with it. The predictions are in terms of constraints on region adjacencies across boundaries, and the shape and attributes, such as color contrasts, of the boundaries themselves. The horizon line constraints are that it will tend to have smoothly changing orientation and be adjacent to a large homogeneous region (the sky). In general, the predicted features are described with constrained attributes determined from the visibility components of schemas.

Figures 6-6 and 6-7 show some of the contour related structures in the initialized PSDB. Figure 6-6 shows the edges extracted at one spatial resolution using the Canny edge operator [Canny - 83]. We have found it useful not to apply noise suppression to extracted segments in order to base filtering on structural properties of the contours, including linear deviation and relationships to other image structures. Different linear segment fits for this extracted edge images are shown in Figure 6-7.

Figure 6-8 shows the results of grouping processes applied to a set of selected curves in Figure 6-3 with multiple associated attributes for orientation and color contrasts. The grouping processes were constrained by the predicted segmentation in Figure 6-5 using constraints on allowable color contrasts, changes in linear segment orientation, and rough image position and extent. Multiple groups are obtained for each predicted image event. Selection of one, or maintaining multiple alternative groups, is explicitly represented in the schema instantiation structure. Here, groups were selected based upon length and uniformity of composite attributes.

Figure 6-9 shows the results of a road schema instantiation based upon matches to extracted road boundaries in accounting for road surface properties through PART-OF relations. Texture elements adjacent to the road boundary which are consistent with a road surface, such as low contrast, parallel edges corresponding to tread marks, are used to direct queries to instantiate potential road area. Queries are also used to determine the presence of anomalous structures in the road such as anything which is high contrast or oriented perpendicular to the road direction. Such structures require disambiguation through instantiation of another schema (it could be a road marking) cued by the anomaly or elevation estimates derived from motion displacements

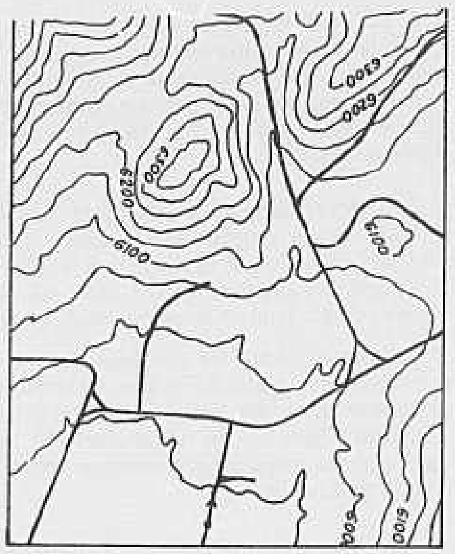


Figure 6-1: Terrain Data

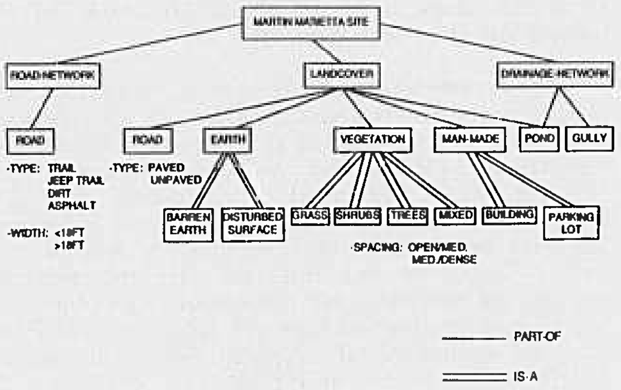


Figure 6-2: A Prior Terrain Type Classification

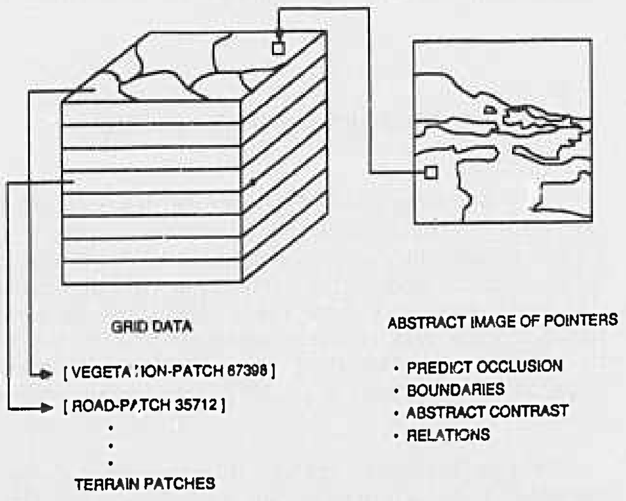


Figure 6-3: Predicted Segmentation From Grid Data

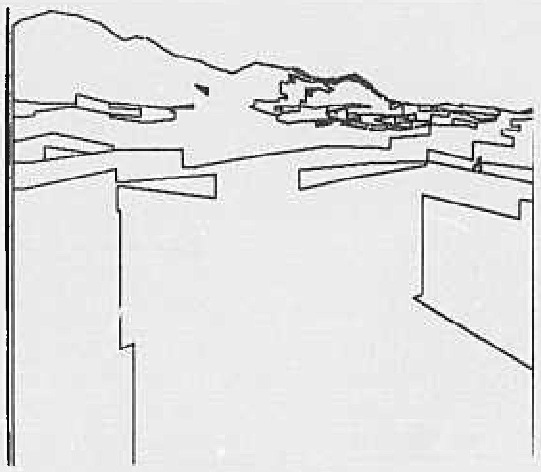


Figure 6-4: Terrain Patches

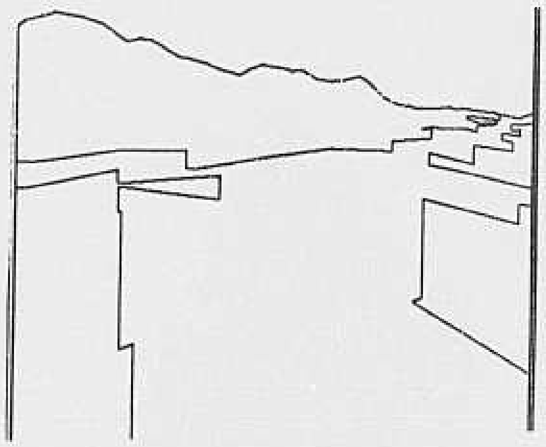


Figure 6-5: Merged Terrain Patches

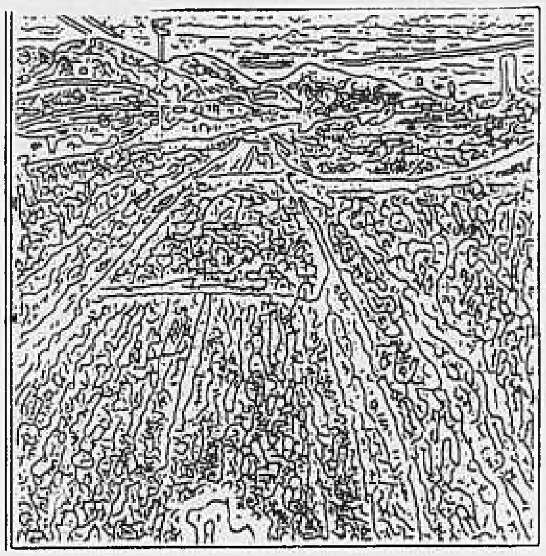


Figure 6-6: Canny Operator

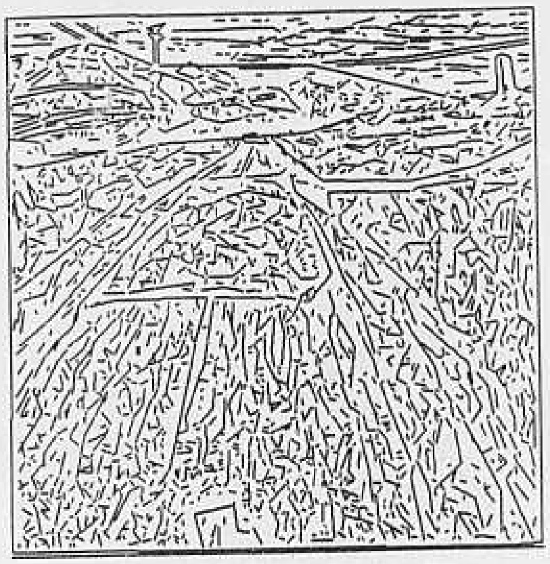


Figure 6-7: Linear Segment Fits

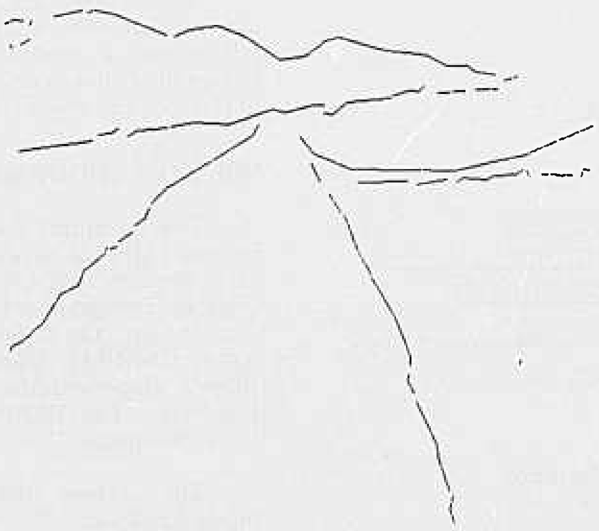


Figure 6-8: Contour Groupings

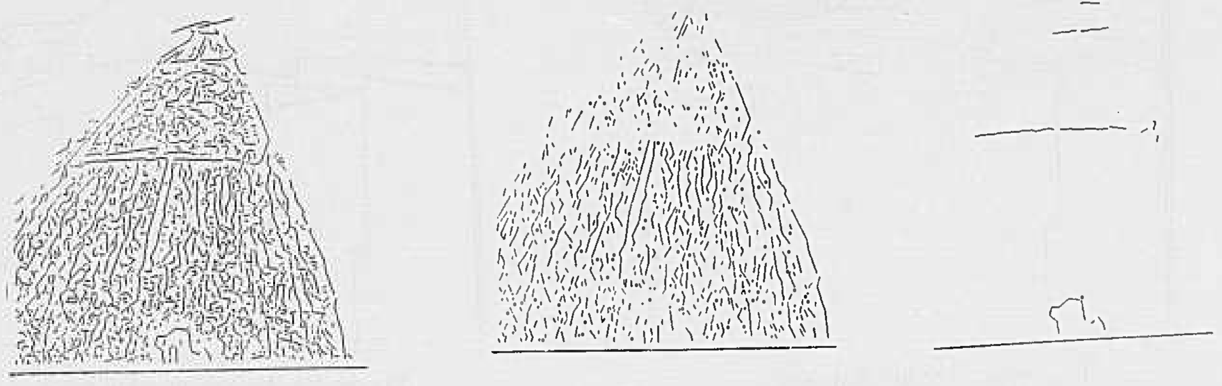
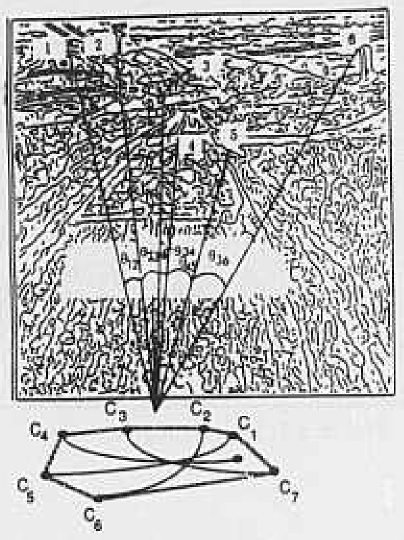


Figure 6-9: Road Schema Instantiation



3D-ANGLES:

$\Theta_{ij} = \cos^{-1}[(\vec{L}_i \cdot \vec{L}_j)/|\vec{L}_i| \cdot |\vec{L}_j|]$
$\Theta_{12} = 10^\circ, \Theta_{23} = 16^\circ, \Theta_{34} = 15^\circ, \Theta_{45} = 11^\circ, \Theta_{56} = 35^\circ$
(assumes $90^\circ \times 90^\circ$ FOV)

RANGES:

PEAK:	2, 4 km	based on LOS correspondence with map
POLE:	.25, .5 km	based on pole recognition, motion stereo
NOTCH:	2, 5 km	based on LOS correspondence with map
TAR:	.1, .5 km	based on flat plane assumption and sensor aspect estimates
INTERSECTION:	.1, .5 km	based on road recognition and map correspondence
BUILDING:	2, 5 km	based on recognition, a priori model knowledge

LOCAL-COORDINATE-SYSTEM: {POLE, NOTCH, INTERSECTION, BUILDING}

PERCEPTUAL-LOCALIZATION-REGION: $\{C_1, C_2, C_3, C_4, C_5, C_6, C_7\}$

Figure 6-11: Viewframe Instance

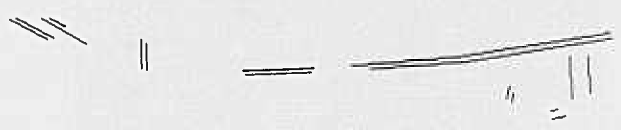


Figure 6-10: Significant Perceptual Groups

7. SUMMARY

The architecture we have developed, using terrain and road schemas with implemented system components for perceptual processing and manipulating long term terrain data, has been successfully used in tasks for ALV navigation and scene interpretation.

ACKNOWLEDGEMENTS

This document was prepared by Advanced Decision Systems (ADS) of Mountain View, California, under contract number DACA76-85-C-0005 for the U.S. Army Engineer Topographic Laboratories (ETL), Fort Belvoir, Virginia, and the Defense Advanced Research Projects Agency (DARPA), Arlington, Virginia. The Contracting Officer's Representative at ETL is Ms. Rosalene M. Holecheck. The DARPA point of contact is Lt. Col. Robert Simpson.

The authors wish to thank Jeanne Gadol and Angela Erickson.

REFERENCES

[Brooks - 84] - R.A. Brooks, "Model-Based Computer Vision", Computer Science: Artificial Intelligence, No. 14, UMI Research Press, 1984.

[Canny - 83] - J. Canny, "A Variational Approach to Edge Detection", In Proceedings of the National Conference on Artificial Intelligence (AAAI-83), pp. 54-58, August 1983.

[Cao et.al. - 86] - Z. Cao, S. Oh, and E. Hall, "Dynamic Omnidirectional Vision for Mobile Robots", Journal of Robotic Systems, Vol. 3, No. 1, 1986, pp. 5-17.

[Hanson et.al. - 78] - A.R. Hanson and E.M. Riseman, "VISIONS: A Computer System for Interpreting Scenes", In Computer Vision Systems, Academic Press, 1978.

[Kuan - 84] - D.T. Kuan, "Terrain Map Knowledge Representation for Spatial Planning", 1st National Conference on AI Applications, pp. 578-584, 1984.

[Kuipers - 82] - B.J. Kuipers, "Getting the envisionment right", In Proceedings of the National Conference on Artificial Intelligence (AAAI-82), Pittsburgh, Pennsylvania, August 1982.

[Levitt et.al. - 87] - T. Levitt, D. Lawton, D. Chelberg, and P. Nelson, "Qualitative Navigation", Defense Advanced Researon Projects Agency Image Understanding Workshop, Los Angeles, California, February 23-25, 1987.

[Marr - 82] - D. Marr, "Vision", W.H. Freeman, San Francisco, 1982.

[Martelli - 76] - A. Martelli, "An application of heuristic search methods to edge and contour detection", Commun. ACM, Vol. 19, No. 2, February 1976, pp. 73-83.

[Ohta - 80] - Y. Ohta, "A Region-Oriented Image-Analysis System by Computer", Ph.D. Thesis, Kyoto University, Department of Information Science, Kyoto, Japan, 1980.

[Pentland - 83] - A. Pentland, "Fractal-Based Description of Natural Scenes". In Proceedings Image Understanding Workshop, Arlington, Virginia, June, 1983, pp. 184-192.

Honeywell Progress on Knowledge-Based Robust Target Recognition & Tracking

Bir Bhanu, Durga Panda and Raj Aggarwal

Honeywell Systems & Research Center
3660 Technology Drive, Minneapolis, MN 55418

ABSTRACT

In the Honeywell Strategic Computing Computer Vision Program, we are working on demonstrating knowledge-based robust target recognition and tracking technology. This report summarizes the progress made during the period July 1986 to January 1987. The focus of our work has been to use artificial intelligence techniques in computer vision, spatial and temporal reasoning and incorporation of a priori, contextual and multisensory information for dynamic scene understanding. The topics currently under investigation are: 1) Landmark and target recognition using multi-source a priori information, 2) Robust target motion detection and tracking using qualitative reasoning, 3) Interpretation of terrain using symbolic grouping. Results from these activities are presented. They are useful in vision controlled navigation/guidance of the autonomous land vehicle, reconnaissance, surveillance, photo-interpretation, and other practical military applications such as search and rescue and targeting missions.

1. INTRODUCTION

The goal of our research in Strategic Computing Computer Vision Program is to demonstrate that knowledge-based approaches as applied to the real-world computer vision problems, such as recognition and tracking of targets and interpretation of terrain, can provide significantly enhanced and robust performance. The results from our research are useful in vision controlled navigation/guidance of Autonomous Land Vehicle (ALV), reconnaissance, surveillance and other practical military applications.

To achieve our goal, we are engaged in developing new techniques for qualitative motion understanding, scene and object modeling, matching, spatial reasoning for recognition, reasoning under uncertainty, symbolic grouping for the interpretation of multi-spectral terrain data, geographic knowledge representation, etc.

Since the knowledge-based techniques, which are being developed here will be a part of a larger system where real-time considerations are very crucial, we are using Real Time Blackboard Architecture (RTBA) software developed at Honeywell.[15] It is written in LogLisp, a logic programming system.[11] LogLisp is composed of Common Lisp and extensions to Common Lisp for logic procedure language. Since the blackboard and LogLisp are written in Common Lisp, RTBA can be used on any hardware/operating system configuration that supports Common Lisp. Currently implemented systems are symbolics 3670 and DEC Vax 11/780 running BSD 4.3 Unix. RTBA is similar to Local Map Builder (LMB) developed at Carnegie Mellon University. However, it is directed towards defense applications where the interest is in building embedded expert systems to achieve real-time performance, to integrate expert system and host system, to establish reliability and to maintain correct inference across changing input data.

The research results described in this report are partitioned into three topic areas: (a) Landmark and Target Recognition, (b) Target Motion Detection and Tracking, and (c) Interpretation of Terrain. We also discuss the applications of this work to Brilliant weapons.

2. LANDMARK AND TARGET RECOGNITION

An autonomous land vehicle has to traverse long distances to accomplish missions such as surveillance, search and rescue and munitions deployment. This results in the accumulation of significant amount of positional error in the land navigation system. Landmark recognition can be used to reduce this error by recognizing the observed objects in the scene and associating them with the specific landmarks in the geographic map knowledge base. In the current ALV test sites at Martin Marietta, Denver, the landmarks of interest include telephone poles, storage tanks, buildings, houses, gates, etc.

We have developed a new approach, called PREACTE (Perception-REasoning-ACTion and Expectation), for using knowledge-based landmark recognition for the purpose of guiding ALV. It is based on the perception-reasoning-action and expectation paradigm of an intelligent agent. This paradigm is different from the test-hypothesize-act cycle of Matsuyama and Hwang.[18] PREACTE uses expectations and allows the prediction of appearance and disappearance of objects in the field-of-view and therefore, it reduces the computational complexity and uncertainty in labeling objects. The approach makes use of extensive map and domain dependent knowledge in a model-based scheme.[10] Our map representation relies heavily on declarative and explicit knowledge instead of procedural methods on relational databases.[19] Davis[12] at Yale University has worked on the problem of acquiring geographic knowledge by a mobile robot. In contrast to his work, in our research explicit knowledge about the map and landmarks is assumed to be given and it is represented in a relational network. It is used to generate an Expected Site Model (ESM) for search delimitation, given the ALV location and its velocity. Landmarks at a particular map site have their 3-D models stored in heterogeneous representations such as generalized cylinder or wire-frame.[6] The landmark recognition vision system for PREACTE generates a 2-D and partial 3-D scene model from the observed scene. The ESM hypothesis is verified by matching it to the image model. The matching problem is solved by using object grouping and spatial reasoning. Unary, binary and ternary relations are used for spatial reasoning. Both positive and negative evidences are used in spatial reasoning and updating the positional uncertainty of ALV in the map. Evidence accumulation is accomplished by an extension of an efficient heuristic Bayesian formula. A common framework for abstracting image information and modeling objects in heterogeneous representations provides a flexible and modular computational environment for reasoning about image content. The system also provides feedback control to the low-level processes to permit adaptation of the feature detection algorithms parameters to changing illumination and environmental conditions. Currently the landmark recognition scheme assumes that the landmarks are along the sides of the road. In the future we will extend it to a more general and complex situation where the ALV may be traveling through terrain and it has to determine precisely where it is on the map by using landmark recognition.

Details of the landmark recognition system PREACTE together with results on ALV imagery are given in.[23]

3. TARGET MOTION DETECTION AND TRACKING

Successful operation of an autonomous land vehicle requires continuous interpretation of complex dynamic scenes utilizing multiple sources of visual and a priori information. Concepts from static scene analysis such as segmentation, feature extraction, spatial grouping and object recognition must be complemented by processes which deal with the temporal aspects of visual perception. Since the ALV moves through a 3-D environment, the resulting camera image is subject to continuous change and objects in the scene cannot be labeled as stationary or moving by simple 2-D techniques. Extensive work has been done in low-level (pixel based) motion analysis,[26] e.g. the computation of optical flow and displacement fields, as well as reconstructive bottom-up approaches to determine 3-D structure and motion. Although some problems remain unresolved and researchers have often made unrealistic assumptions such as orthographic projection, stationary viewer or static environment, the field is well developed and useful techniques are available.[21] However, the integration of motion information into the higher levels of vision is still in its infancy. The integration requires techniques for knowledge representation and reasoning about events in space and time. This capability is crucial to the navigation of an ALV in unstructured environments.

Dynamic scene understanding can be structured into basically three levels of processes. While the *low* and *high* levels have their equivalents in many other vision approaches, the important role of *intermediate-level* processes has not been clearly identified and defined. Low-level processes are purely two-dimensional, bottom-up and image-centered, such as feature extraction, feature matching or optical flow computation. High-level processes are three-dimensional and world-centered, attempting the semantic interpretation of the dynamic proceedings in the environment, using information about the structure and motion of 3-D aggregates. Note that long-term observation and understanding of the behavior of objects form the basis for intelligent actions, such as navigation, route planning and threat handling. In order to bridge the representational gap between low and high levels, we introduce processes operating at an *intermediate level*, characterized by the transition from image-centered features to world-centered objects. Unlike at the low and high levels, bottom-up and top-down strategies combine at the intermediate level and 3-D reasoning is supported by aggregation of physical and perceptual knowledge and expectations. In psychological terms this stage could be related to unconscious but active visual perception; those tasks that are performed continuously and automatically by the human visual system.

We have developed a new DRIVE (Dynamic Reasoning from Integrated Visual Evidence) approach based on a *Qualitative Scene Model* to solve the motion understanding problem. The approach addresses the key problems of the estimation of vehicle motion from visual cues, the detection and tracking of moving objects and the construction and maintenance of a global dynamic reference model. Object recognition, world knowledge and accumulation of evidence over time are used to disambiguate the situation and continuously refine the global reference model. The approach departs from previous work by emphasizing a qualitative line of reasoning [13] and modeling, where multiple interpretations of the scene are pursued simultaneously in a hypothesis and test paradigm. Different sources of visual information such as two-dimensional displacement field, spatial reasoning and semantics are integrated in a rule-based framework to construct and maintain a vehicle centered three-dimensional model of the scene. This approach offers significant advantages over "hard" numerical techniques which have been proposed in the motion understanding literature.[1,20,27] These advantages include the tracking of objects in the presence of partial or total occlusion and use of this information for route planning and threat handling.

In the DRIVE approach a vehicle-centered model of the scene is constructed and maintained over time, representing the current set of feasible interpretations of the scene. In contrast to most previous approaches, no attempt is made to obtain an accurate geometric description of the scene. Instead a *Qualitative Scene Model* is proposed which holds only a coarse qualitative representation of the three-dimensional environment. As part of this model, the "stationary world" is represented by a set of image locations, forming a rigid 3-D configuration which is believed to be stationary. All the motion-related processes at the intermediate level use this model as a central reference. The motion of the vehicle, for instance, must be related to the stationary parts of the environment, even if large parts of the image are in motion. This kind of reasoning and modeling appears to be sufficient and efficient for the problem at hand.

Details of the qualitative reasoning concept emphasizing the motion aspects of intermediate-level processes and interfaces to the adjacent levels in the DRIVE system are presented in.[8]

4. INTERPRETATION OF TERRAIN

An autonomous land vehicle must be able to navigate not only on the roads, but also through terrain in order to execute its missions of surveillance, search and rescue and munitions deployment. To do this the vehicle must categorize the terrain regions it encounters as to the trafficability of the regions, the land cover of the regions and region-to-map correspondence.

Predominantly, the segmentation algorithms used for outdoor scene segmentation are region analysis algorithms [22,24,25] which attempt to identify regions of the scene on the basis of the homogeneity of the region's features. Recursive segmentation based on the analysis of distribution of features is one of the most popular and commonly used techniques for image segmentation. Many of these techniques make use of an elaborate peak location and selection procedure which provides threshold values for the purpose of image segmentation. The computation of peak maxima and minima is complicated since minor changes must be distinguished from major ones. One of the shortcomings of these techniques is that small regions in a large image may not show a distinct peak in the histogram, even if they are distinct from their immediate neighborhood. Therefore, in the application of these techniques, normally the image is partitioned artificially into a set of subimages and each subimage is segmented and split further independently. As a result, a remerging measure may be required to merge regions that are arbitrarily broken at the subimage boundaries. Very often this merging step leads to some regions which remain unmerged or overmerged.

Parvin and Bhanu[25] have presented a simpler and computationally efficient technique which does not have the above disadvantages and provides good results. It is based on the generalization of a two-class gradient relaxation algorithm for the segmentation of natural scenes.[5]

However, since the outdoor scenes in the ALV scenario have immense variability, purely region based segmentation algorithms do not perform adequately, because they fail to integrate constraints derived from the three-dimensional attributes of the scene and other auxiliary data into the segmentation process such as the information from a standoff-sensor. Scene variability leads to poorly defined region boundaries and spurious noise regions in the segmented image and this degrades the performance of high-level region labeling schemes which operate on these low-level results. Also the unstructured nature of the outdoor scenes makes their segmentation very difficult with a single set of rules. Currently very simple segmentation methods are used for road-following which are severely limited in robustness and flexibility.[17]

The use of three-dimensional qualities for segmentation is critical for the ALV scenario because the range varies significantly with respect to scan line of the image and feature measures which are valid for a specific range may produce unsatisfactory results at other ranges. Also it is very important that we make use of the spectral properties of the objects in the world and a priori and contextual information to achieve robust interpretation of terrain in a flexible manner.

Our approach for terrain interpretation employs a hierarchical region labeling algorithm for ERIM 12 channel Multi-Spectral Scanner data. The technique called, HSGM (Hierarchical Symbolic Grouping for Multi-spectral data), is specifically designed for multi-spectral imagery, but is appropriate for other categories of imagery as well. For this approach, features used for segmentation vary from macro-scale features at the first level of the hierarchy to micro-scale features at the lowest level. Examples of labels at the macro label are sky, forest, field, mountain, road, etc. For each succeeding level of the hierarchy, the identified regions from the previous stage are further subdivided, if appropriate, and each region's labeling is made more precise. The process continues until the last stage is reached and no further subdivision of regions from the preceding stage appears to be necessary. Examples of region labels for this level of the hierarchy are gravel road, snowberry shrub, gambel oak tree, rocky ledge, etc.

The HSGM approach is distinct from the classical tree classifier approaches used in the remote sensing literature. The approach operates as follows: For the first stage of the hierarchy, each of the 12 channels of the Multi-Spectral Scanner data is segmented with a textured region detection scheme. The individual segmentations for the 12 channels are combined by using a edge linking relaxation operator[9] to define a "plan" region boundary image. Representative features for each region of the "plan" image are calculated by averaging the feature values for each pixel of the region across the entire region area. These features are classified with a rule-based classification scheme which uses contextual as well as spectral cues for region labeling. Then, at the succeeding stages of the hierarchy, the regions are subdivided by a variety of region and edge-based segmentation algorithms which are optimized for the specific category of region under consideration. These segmentation algorithms also employ spectral, contextual and auxiliary information cues for the specification of region boundaries. Examples of the applied constraints for these stages are a priori terrain elevation data, land cover map information, geological data, time of day and seasonal information. HSGM approach possesses several attractive features, the most important of which is its robustness in the presence of high scene variability. Because the finalized region classifications are derived with global support, both from neighboring regions and from other spectral images, the likelihood that a region will be misclassified because of arbitrary noise is greatly reduced. This approach is also computationally less expensive than many rule-based region labeling schemes because the application of auxiliary constraints decreases the branching factor of the search process significantly.

Details of the HSGM technique with initial results and examples from real ALV imagery are given in.[9]

5. BRILLIANT WEAPONS APPLICATIONS

In addition to the ALV applications as discussed in the above, our interest is also to transfer this technology to other practical military applications. Precision Guided Weapons (PGWs) are one such application. Conventional technology such as Automatic Target Recognition (ATR) has come a long way but it needs help.[7] It is clear that for the vision technology to succeed in practical brilliant weapons application, it must be optimaly suited for such multisensor combinations as millimeter wave/infrared,[2,4,16] and CO_2 laser.[14] One of our objectives is to transfer the knowledge-based technology under development here to brilliant weapons relevant multisensor applications to provide significant improvement in performance in diverse scenarios, especially in inclement weather and battlefield scenarios. The vision system performance must demonstrate robustness in hundreds of hours of classified flight test sensor data in presence of target camouflage, concealment and deception (CCD). We are using multisensory and a priori information in a knowledge-based framework within RTBA to achieve the required performance which is beyond what conventional ATR technology can provide.[3]

6. CONCLUSIONS

In this report we have presented a summary of our work completed during the last seven months. In the future we plan to integrate our PREACTE module for landmark and target recognition with DRIVE module for qualitative motion understanding and HSGM module for terrain interpretation for an end-to-end simulation demonstrating knowledge-based scene dynamics approach for target motion detection, recognition and tracking.

REFERENCES

1. *Proc. IEEE , Workshop on Motion: Representation and Analysis, Kiwah Island Resort, Charleston, South Carolina.* May 7-9, 1986.

2. *Dual-Mode Seeker Study, Final Report, Airforce/Army Contract No. F08635-85-C-0156, Honeywell Systems & Research Center.* May 1986.

3. *Smart Weapons Program, DARPA/U.S. Army AMCCOM Armament Research & Development & Engineering Center, Contract no. DAAA21-86-C-0305, Honeywell Systems & Research Center.* 1986.

4. W. Au, S. Mader, and R. Whillock, "Scene Analysis," Third Triannual Technical Report, Night Vision & Electro-Optics Lab, Contract No. DAAL01-85-C-0429, Honeywell Systems & Research Center (July 1986).

5. B. Bhanu and O.D. Faugeras, "Segmentation of Images Having Unimodal Distributions," *IEEE Trans. on Pattern Analysis and Machine Intelligence* **PAMI-4**(4) pp. 408-419 (July 1982).

6. B. Bhanu and T. Henderson, "CAGD Based 3-D Vision," Proc. IEEE International Conference on Robotics & Automation (March 1985).

7. B. Bhanu, "Automatic Target Recognition: State of the Art Survey," *IEEE Trans. on Aerospace & Electronic Systems* **AES-22**(4) pp. 364-379 (July 1986).

8. B. Bhanu and W. Burger, "DRIVE: Dynamic Reasoning from Integrated Visual Evidence," Proc. DARPA Image Understanding Workshop (These Proceedings) (Feb. 1987).

9. B. Bhanu and P. Symosek, "Interpretation of Terrain Using Hierarchical Symbolic Grouping for Multi-Spectral Images," Proc. DARPA Image Understanding Workshop (These Proceedings) (Feb. 1987).

10. T.O. Binford, "Survey of Model-Based Image Analysis," *The International Journal of Robotics Research* **1** pp. 18-64 (Spring 1982).

11. J. Carciofini, T. Colburn, and R. Lukat, "LogLisp Programming System User's Guide," Internal Report, Honeywell Systems & Research Center (July 1986).

12. E. Davis, *Representing and Acquiring Geographic Knowledge,* Morgan Kaufmann Publishers, Inc. (1986).

13. B. Kuipers, "Qualitative Simulation," *Artificial Intelligence* **29** pp. 298-338 (1986).

14. K. Landman, "Automatic Laser Target Classification," Phase I Interim Report, AFWAL/Avionics Lab Contract No. F33615-84-C-1519, Honeywell Systems & Research Center (April 1985).

15. A. Larson, "Real-Time Blackboard Architecture System," Internal Report, Honeywell Systems & Research Center (August 1986).

16. B. Lee, W. Higgins, and J. Gillberg, "Multisensor Algorithm Development: First - Fourth Quarterly Reports," Night Vision and Electro-Optics Lab, Contract no. DAAL 01-85-C-0443, Honeywell Systems & Research Center (1986).

17. J. Lowrie, "The Autonomous Land Vehicle Second Quarterly Report," Martin Marietta, Denver, Colorado (September 1986).

18. T. Matsuyama and V. Hwang, "SIGMA: A Framework for Image Understanding - Integration of Bottom-up and Top-Down Analysis," Proc. Int. Joint Conference on Artificial Intelligence (August 1985).

19. D.M. McKeown, Jr., W.A. Harvey, Jr., and J. McDermott, "Rule-Based Interpretation of Aerial Imagery," *IEEE Trans. on Pattern Analysis and Machine Intelligence* **PAMI-7** pp. 570-585 (September 1985).

20. G. Medioni and Y Yasumoto, "Robust Estimation of 3-D Motion Parameters from a Sequence of Image Frames Using Regularization," Proc. DARPA Image Understanding Workshop (Dec. 1985).

21. H.-H. Nagel, "Image Sequences - Ten (octal) Years - From Phenomenology Towards a Theoretical Foundation," Proc. International Conference on Pattern Recognition (October 1986).

22. P. Nagin, A. Hanson, and E. Riseman, "Studies in Global and Local Histogram Guided Relaxation Algorithms," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pp. 263-277 (May 1982).

23. H. Nasr, B. Bhanu, and S. Schaffer, "Guiding the Autonomous Land Vehicle Using Knowledge-Based Landmark Recognition," Proc. DARPA Image Understanding Workshop (These Proceedings) (Feb. 1987).

24. R. Ohlander, K. Price, and D.R. Reddy, "Picture Segmentation Using a Recursive Region Splitting Method," *Computer Graphics and Image Processing* **8** pp. 313-333 (1978).

25. B. Parvin and B. Bhanu, "Segmentation of Natural Scenes," *Pattern Recognition, in Press*, (1987).

26. K. Prazdny, "On the Information in Optical Flows," *Computer Vision, Graphics and Image Processing* **22** pp. 239-259 (1983).

27. E.M. Riseman and A.R Hanson, "Summary of Progress in Image Understanding at the University of Massachusetts," Proc. DARPA Image Understanding Workshop (Dec. 1985).

SECTION II

TECHNICAL REPORTS

PRESENTED

THE ITA *RANGE IMAGE PROCESSING SYSTEM*

David G. Morgenthaler, Keith D. Gremban*, Mitch Nathan, John D. Bradstreet**

Martin Marietta Denver Aerospace

P.O. Box 179, M.S. H0427

Denver, CO 80201

Abstract

Under the Intelligent Task Automation program a vision system for processing range image data was developed for use in a robotic inspection scenario. The objectives for the vision system in this program were to use the range data sensor as both a measurement sensor and as a vision sensor for robotic guidance. The chosen sensor is a time of flight sensor developed at ERIM. This paper is intended to be an overview of this range image processing system and its use together with the robotic system for inspection. This paper describes the design and operation of the vision system as both a measurement tool and as an object recognition component of the system. Examples of the system operation are given, as well as a discussion of potential uses of the system for robotic assembly and manufacturing.

1. INTRODUCTION

The objectives of the Intelligent Task Automation program were to develop generic technologies which have near-term applicability to batch manufacturing and long-term potential for complex military tasks, and to demonstrate the feasibility of these selected generic technologies. We identified three technology areas as critical to the development of advanced robotic systems: offline programming, visual sensing, and servo-controls. We chose as our demonstration implementation a system for performing dimensional inspections of a F-15 bulkhead. This paper describes the visual sensing technologies developed under this program.

The objectives for the vision system design were to achieve easy programmability of the vision system for other tasks, and to achieve competence in unstructured environments. While the speed limitation intrinsic to vision systems can be mitigated through proper design and implementation of the software subsystem, it is more appropriately addressed by hardware engineering and was not fully addressed.

By the programmability of a vision system for other tasks we mean the ability to easily reprogram the vision system to accomplish these other tasks. When vision tasks are represented implicitly through the procedural definition of the software coding, modification of the vision system behavior, either by the vision system itself or by a human, is difficult because the modification (re-programming) is non-procedural. Our approach to realizing this objective is to organize low and intermediate image processing tasks around geometric and topological constructs which are independent of objects, and to use CAD/CAM-like models to provide object specific information for recognition. Task specific information is input through the offline programming subsystem and operated upon by the planning subsystem.

Objects to be perceived by the vision system have an explicit viewpoint independent representation called the object-model within the system. When an unexpected visual situation is presented (e.g., due to a different orientation or viewpoint to the object) the system can easily adapt because its operation is independent of any particular visual situation. When a new object or objects constitute the domain of the visual tasks no reprogramming of the system is needed to recognize these new objects; only the object-models and the measurement abstractions of interest (e.g., what is the length of the object?) need to be specified.

By competence in unstructured environments we mean the ability of the vision system to accomplish its task reliably in all of the many visual situations which may be presented by an object and its environment. Our

*Current address: Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA 15213.

**Current address: Computer Science Department, University of Colorado, Boulder, CO.

The support of the Defense Advanced Research Projects Agency and the Air Force Wright Aeronautic Laboratories under contract F33615-82-C-5139 is gratefully acknowledged.

goal here is to not resort to the usual structuring of the environment in which the visual situation is restricted by providing specialized lighting (e.g., lighting from behind the object to provide a high contrast silhouette) or by forcing a limited number of viewpoints of the object (e.g., fixturing the objects to rest on only one of their sides, and possibly without varied rotational orientation).

The vision system described in this paper achieves some degree of freedom from these environmental constraints through the use of an active laser range imaging sensor. Because the sensor is active it is relatively insensitive to prevailing lighting conditions. Because the sensor provides range measurements from the closely spaced points on the object it is possible to recover through various processes the object's three-dimensional orientation and overall distance. These sensor characteristics, coupled with the object-model methodology of the system, achieves competence within a great variety of environmental visual situations.

1.1. References and Related Research

The laser range imagery used in this research is obtained from a scanning laser ranger operating on a time of flight principle. Another ranging technique which is even beginning to appear in commercially available vision systems is structured lighting[1–2]. The principle advantages of these approachs are the ability to recover three-dimensional information quickly, and the relatively simple and inexpensive apparatus required. Additionally, because it can use a standard video camera it may be used in conjunction with other video image processing techniques (e.g., connected component analysis of high contrast images). Its primary disadvantages are that it produces only sparse range data, and that because of occlusion it may not be possible to determine ranges to certain portions of a scene within the image. The range of distances over which this technology can be used is limited by practical separation distances of light source and sensor.

Other approaches to obtaining range data are the growing number of "shape-from-X" techniques[3–7] , where for "x" we may substitute "stereo", "shading", "texture", "optical flow", and even "shape", to name a few. In part the premice of our work is that these techniques will someday be found to be computationally practical; thus we should be prepared with algorithms to process and interpret the dense range data produced by the shape-from-x techniques.

We also proposed to use object-modeling techniques[8–11] to help achieve our objectives. In our approach models are viewpoint independent and are used only in a bottom up fashion and at the uppermost level of processing. A complete scene model is created from the range image without reference to the model. Interpretation of the scene model is accomplished by a matching process which matches scene models to *a priori* object-models in a knowledge base.

Additional details of the ITA program research and a fuller list of references can be found in [13].

2. LASER RANGE IMAGING SENSOR

The laser range imaging sensor produces digital images where the image value or greylevel at each image point represents the range from the sensor focal point to the corresponding point in the visual scene. Details of the operation of the laser range imaging sensor used may be found in [12]; a summary description of its operation and imaging characteristics is given here.

The sensor operation is based on time of flight, but rather than measuring elapsed time (which would be much to short for the distances involved) the sensor measures the phase shift between outgoing and returning modulated signals. A laser diode source is amplitude modulated to one of two carrier frequencies, and transmitted though appropriate optics and mirrors to the target. A returning waveform is received, and after amplification is compared with the transmitted waveform to determine the phase shift. This phase shift is measured to 8-bit resolution and is directly proportional to the range to the target modulo the period of the waveform. This period of the waveform is called the ambiguity interval length. The ambiguity length can be halved by use of a phase multiplier.

To these range sensing electronics are added computer controlled optics and beam steering mirrors. The computer controlled optics change the total possible field of view from approximately 1m by 1m at 1m for the low resolution modes to approximately 8cm by 8cm at 20cm for the high resolution modes (see below). The beam steering mirrors are software controllable so that arbitrary scan patterns can be obtained. If the full field of view were raster scanned without over- or under-sampling image sizes of approximately 3000x3000 pixels would be obtained. It is possible to cause a raster scan of the scene such that a 256 x 256 pixel image can be acquired in about 0.5 seconds.

By varying the carrier frequency and optionally using a phase multiplier it is possible to acquire range data with four different resolutions. Table 2-1 summarizes the four modes of data capture with respect to resolution, accuracy, and ambiguity interval length. Under computer control it is possible to switch between these modes with a delay of 22 msec, so that it is possible to produce 16-bit range data by combining the measurements made at several different resolution levels.

Because the mirror movement used to scan the beam is computer controlled it is also possible to control the density of sample points. Thus, it is possible to undersample (for example by acquiring .002 in. resolution measurements at 0.02 in. spacing) or to oversample (for example by acquiring 0.002 in. resolution measurements at 0.001 in spacing).

Table 2-1

LASER RANGE IMAGER			
OPERATIONAL CHARACTERISTICS			
Mode	Resolution	Accuracy	Ambiguity Interval Length
1	0.0005 in.	0.001 in.	0.128 in.
2	0.002 in.	0.002 in.	0.512 in.
3	0.032 in.	0.032 in.	8.192 in.
4	0.128 in.	0.128 in.	32.77 in.

The largest possible field of view is 20 degrees in modes 1 and 2, and 50 degrees in modes 3 and 4. Although it is possible to acquire range data throughout the entire field of view while neither oversampling nor undersampling, this would lead to intractably large images. Instead, the usual practice is to acquire images within a smaller field of view without over- or undersampling; the largest possible field of view is called the field-of-regard. Thus several different images can be obtained within the field-of-regard without moving the sensor.

3. OPERATION OF THE VISUAL INSPECTION SYSTEM

3.1. Modes of operation

The vision system described in this paper can be used in two ways to perform dimensional inspections in a robotic or other highly automated manufacturing process. When the accuracy of the vision system is adequate for the specific dimensional measurement it is used directly to perform the measurement. When greater accuracy is needed, or when it is difficult to position the sensor for the measurement, the system is used in conjunction with robotic manipulators to guide the placement of other inspection tools.

The system described in this paper was developed for use in a robotic system where it would function in both of these capacities. Specifically, the robotic system is to perform the dimensional measurements on an F-15 bulkhead as illustrated in Fig. 3.1-1. Dimensional measurements are made directly with the laser range imaging sensor and associated software when the required accuracy of these measurements is not greater than 0.001 in., and the entire dimensional measurement can be viewed within one field of regard. For these measurements the laser range imaging sensor is held by the robotic manipulator for positioning to make the proper measurement.

When the accuracy requirements are too severe, or the magnitude of the dimensional measurement is so large as to exceed any field of regard of the sensor, the vision system is used to provide visual feedback to the robotic controller to aid in the positioning of another measurement tool. Other measurement tools available include ultrasonic thickness sensors, digital calipers, and even roughness sensors.

3.2. Processing Overview

Whether the vision system is used directly for measurement or used for robotic control the first steps of the visual processing are the same. That is, before a measurement can be made, or before the vision system can provide trajectory information to the robotic controller, the vision system must solve the problem of object identification and location. By object identification we understand that the visual scene contains the object as well as fixtures, parts, and apparatus, and that it is necessary to identify those portions of the image which correspond to the object to be measured.

Further, it is necessary to correctly form the correspondences between the various features of the object (e.g., edges, faces) and the images of these features. By object location we understand that the object may assume any three-dimensional orientation and overall distance from the sensor, and that it is necessary to recover these from the image.

Once these preliminary steps of recovering object identification and orientation have been successfully performed operation of the system can proceed along one of two paths according to the type of inspection to be performed. When the dimensional measurement can be made directly by the range sensor the correspondence between features of the object and the image of these features is used to effect the measurement.

For example, suppose it is desired to measure the distance between two different edges of the object. Since the correspondence between these edges and the images of these edges is known, it is possible to acquire imagery at the required aeeuraey of these edgcs, and from the three-dimensional data obtained to eompute the distanee between these edges. In a similar fashion it is possible to mcasurc the anglc between two faees, curvatures of fillcts, and hole sizes.

If the vision system is to be used to guidc the plaeement of anothcr measurement dcviee by a robotie manipulator then the gross distanee data together with the three-dimensional orientation data is used by the robotic eontroller to eompute an approach trajectory. The vision system can be used to locate both the object to be measured and the measurement tool. By providing highly aceuratc dcseriptions of the position of thc robotie end-effector relative to the object, high absolute aeeuracy of the robotie manipulator is not required.

4. OBJECT IDENTIFICATION AND LOCATION OVERVIEW

In order to solvc thc problems of object idcntifieation and location the systcm performs three distinct types of processing on the range data input. The first type of process, performed on thc raw range data input, consists of low lcvcl image proccssing to recover gcometric scene elements. The second type of processes operate on a eombination of processed imagcry data and low level symbolic data to recovcr topologieal relations among the geometric elements. The third type of process operates entirely on symbolic data to cstablish the seene element to model elemcnt relations.

The processing performed by eaeh type of proeess will be best understood if the overall plan and flow of the system is first understood. In this section an overview of the operation of thc vision system is given. Subsequent sections provide detailed presentations of one or more modules from each process type.

The vision systcm has becn designed so as to explicitly scparatc the proccsses of object identification and location from the shapes and dcseriptions of the objccts. To aeeomplish this any object is eonsidered to eonsist of a number of geometrie clements (i.e., planar faees, straight line cdgcs, vertiees, simple curved surfaees, eurvcd edges, etc.) whieh are connected together aeeording to the topology of the object while maintaining eertain geometrie relationships among these clements (c.g., planes forming right angles). Descriptions of evcry object to be eonsidered by the vision system must be eneoded aeeording to this seheme and stored in a data base; the deseription eorresponding to each objcct is ealled its object-model. The exaet form and use of this objeet-model data base will be diseussed below.

The mcthodology of the vision system is to first proeess the rangc data obtained from the range imaging sensor to segmcnt thc image into geometrie elemcnts whose three-dimensional location and orientation are eomputed. The seeond stcp is to to recovcr three-dimensional topology, and subsequently three-dimensional geometric relationships; image topology is used as a guide together with the three-dimcnsional geometrie element location and orientation data from the first step.

The eombination of these geometric elemcnts found within the image, their three-dimensional topology, and their three-dimensional geometrie relationships togethcr are eallcd the image-model. The final step of visual proeessing for objeet idcntification and loeation is to determine the eorrespondenees between the image-models and objeet-models in the data basc.

When a correspondenee has been found between eertain structures of the image-modcl and an objcct model the object recognition problem has bccn solved. The transformation between the eoordinate system used to deseribe the image-model and the objeet-centered eoordinate system used in the object-model, together with the image-modcl geometrie loeation information provide the solution to thc object loeation problem.

A block diagram for the vision system thus eonstrueted is shown in Fig. 4-1. The laser range imagcry data is proccssed by a numbcr of low level vision processes which idcntify gcometrie features within the image. The output from these geometric processes is of two types. The first is a featurc map, wherein the image area eorresponding to each identified featurc is labeled (or colored) with a unique label. The seeond typc of output from these modules is a property list, wherein for each labcl identifying a feature speeific three-dimcnsional geometry information is given. For example, the output from the planar surfaee deteetor would eonsist of thc surfaee ccntroid and surface normal; surface extent is eontaincd implieitly in the planar surface map image.

After the rangc data imagc has been processed by the geometrie processcs the resulting fcature maps and fcature property lists are proeessed by the topological analysis module. This module recovers the three-dimensional topology of a scene by using the two-dimensional image topology as a guide. By comparing imagc maps of the various features it is simple to detcrmine adjaeeney of the images of these fcatures. Onee image adjacency is found, geometrie analysis will determine if three-dimensional eonneetedness of the fcatures is eonsistent with the images of thc features. For example, two planar surfaecs whieh are adjaeent within the image may be geometrieally intersected, and it may be determined if the image of the line of interseetion eoineides with the adjaceney of the images of the two planes; if so, then the planar surfaees are conneetcd three-dimensionally, and if not then one surfaee oeeludcs the other.

After completion of the topological analysis module the image is represented by a graph-like structure call the image-model. Nodes of this structure represent three-dimensional geometric elements such as planar surfaces, curved surfaces, straight-line edges, curved edges, and vertices. Arcs between nodes represent topological relations which exist between the nodes. Arcs between surface nodes of different types represent connectedness, arcs between surface nodes and edge nodes represent the "is-bounded-by" relation for surfaces, and arcs between edge nodes and vertex nodes represent the "is-bounded-by" relation for edges. Each node explicitly contains the geometric information appropriate to the surface type (e.g., centroid and surface normal for a planar surface), and each arc may have associated with it various geometric information (e.g., where two planar surfaces are connected the angle formed may be specified).

In order to establish correspondence between substructures of the image model and object-models contained in the data base it is useful to perform one additional transformation on both the image-model and on the representation of objects within the data base.

Within the data base we choose to represent objects as a set of statements which form a proposition which may be either true or false. An example of a proposition for the stair-step object is "If there is a trihedral corner formed by some vertices #p1, #p2, #p7, and #p6, such that (#p1,#p2) is a convex edge and (#p1,#p7) is a concave edge and (#p6,#p1) is a convex edge, and if there is a trihedral corner formed by some vertices #p7, #p8, #p1, and #p2, such that...then the vertices #pi form an object called a one_step." In other words, these propositions are clauses in first order predicate calculus where the variables #pi are unbound. The data base is a collection of similar propositions, with one proposition describing each object.

The transformation performed on the image-models is that of interpreting the image-model structure to form assertions of truth in the first order predicate calculus. For example, for each vertex node in the image-model graph structure we can produce a clause naming the edges which terminate at the vertex and, by interrogating the geometric information contained in the structure, can describe those edges as convex or concave. Other clause types are used to describe edges and surfaces.

The problem of object identification then is to prove that one proposition of the object-model data base is in fact a theorem given the assertions of truth made from the image-model. The problem of theorem proving is well understood, and there is a great body of knowledge about how these problems can be most efficiently solved. Details of how this process can be speeded up for object recognition will be discussed below.

5. GEOMETRIC MODULES

As described above, the raw range data from the laser imager is first processed by a number of modules which extract geometric primitives from the imagery. By extraction we mean that the image area corresponding to a three-dimensional geometric shape can be colored with a unique label, and that associated with this label are the parameters which uniquely describe the geometric shape. The visual inspection system currently is capable of extracting the following types of geometric elements from the images: Planar surfaces, second order surfaces, straight line edges, and vertices. In the following sections we describe the various techniques developed for extracting these geometric elements.

5.1. Planar Surfaces

The planar surface analysis process accepts as input a range image, and provides as output a surface property list and a surface map image in which each pixel belonging to a planar surface is labeled with the number of that surface. For each surface the surface property list contains the (x,y,z) location of the centroid, the surface normal, and the smallest window within the image containing the surface.

To understand the operation of the surface detection approach refer to the coordinate system and sensor perspective model of Fig. 5.1-1. The sensor is located at the origin and is boresighted along the z-axis. We assume that the focal length of the sensor f is known or can be computed, and that the z-axis passes through the center of the image.

The azimuth angle θ and the elevation angle ϕ to each pixel in the image can be computed as functions of the row and column locations within the image . Similarly, given a range value at (row, col) for some pixel in the image it is possible to compute the x, y, and z locations to which this refers in the scene; that is, the location of the ray through the image pixel (row, col) of length R, where R is the range given by the image value.

Given a range image R(row, col), the surface normal N_i at a surface point p is computed as the cross product of the two surface tangent vectors

$$\vec{V}_1 = (\frac{\delta x}{\delta row}, \frac{\delta y}{\delta row}, \frac{\delta z}{\delta row}) \; at \; p$$

$$\vec{V}_2 = (\frac{\delta x}{\delta col}, \frac{\delta y}{\delta col}, \frac{\delta z}{\delta col}) \; at \; p$$

where

$$x(\theta,\phi) = R(\theta,\phi)\sin(\theta)\cos(\phi)$$

$$y(\theta,\phi) = R(\theta,\phi)\sin(\phi)$$

$$z(\theta,\phi) = R(\theta,\phi)\cos(\theta)\cos(\phi)$$

Wc have used 5x5 Prewitt masks to estimatc the various partial derivatives

Once thc above calculations are done we are left with a vector of images $\vec{N}_p = \vec{V}_1 X \vec{V}_2$. The next step in planar surface extraction is to collect or aggregate subsets of pixels which correspond to planar surfaces. The planar surface aggregation approach is motivated by the property of planar surfaces in Euclidean and digital geometry that every connected subsct of a plane is co-planar.

The aggregation process is performed by a connected component process which computes thc reflcxive, transitive, symmetric closure of the SIMILAR relation, defined as follows for each image point p and its ncighbor q:

$$SIMILAR\ (x,y) = \begin{cases} 1 & \text{if } \cos^{-1}\left[\dfrac{N_x * N_y}{|N_x||N_y|}\right] > \lambda \\ 0 & otherwise \end{cases}$$

for some threshold λ close to 1. Thus, two points are SIMILAR if thc angle between their normal vectors is small.

At this point each component consists of points that are pairwise approximately planar (to within the threshold). However, the component may not be globally planar, and may instead be a slowly curving surface. Thus, a final global planarity check of each component must be made.

This global check is implemented by using the moments to detect components that are not thin in the direction of one of the eigenvectors of the covariance matrix, and so are not globally planar. Also, components of lcss than a certain size (area) can be deleted from furthcr consideration.

Fig. 5.1-2 shows results obtaincd using this technique for a simple stair-step shaped object. The raw range image is shown in Fig. 5.1-2(a) and the planar surface map is shown in Fig. 5.1-2(b).

5.2. Straight Line Edge Detection

Our straight line edge detection algorithm consists of a edge pixel detection phase followed by an edge aggregation and classification phasc. Each phase is describcd below.

We have dcveloped two approaches to edgc pixel detection. The first approach is variant of the ∇^2 G edge detection schemc[14] . In range images the ∇^2 G zero crossing operator detects only occluding (i.e., step-like) edge pixels in the scene. Extension to a ∇^3 G operator allows detection of concave and convex edge (i.e., roof- or vallcy-like) edge pixels. Convex and concave pixels can be distinguished from one another by the sign of thc ∇^2 G value of the pixel. Thus the first edge pixel detection approach not only detects edge pixels, but can label them by type.

The second edge pixel detection approach is a byproduct of the planar surface dctection approach. It is assumed that the scene consists entirely of smooth surfaced objects. The first step of the planar surface detection algorithm detects smooth surfaces (and latcr determines if they are planar). The remaining non-smooth points are thus candidates for edge and vertex points. This approach does not immediately classify edge pixels by type (convex, concave, occluding), although this classification can be computed later by inspection of the surfaces on either side of an edgc. This approach has the advantage that the edge dctection step is to simply invert the planar surface map.

The second phase of the processing is to aggregate edge pixels into straight line subsets. The algorithm developed for this purpose performs the following tasks:

- Edge pixcls making up intcrsections and noise arc removed. This separates linear connected sets of edges from each other;
- Linear connected sets of pixels are located and parameterized;
- Gaps between collinear segments are filled in and lines are extended by extrapolation from the edgepoints.

The end result of the aggregation process is a linear edge property list and a linear edge map image.

5.2.1. Local Analysis of Linearity

Consider a connected set of pixels that form a straight line (we shall ignore the subtleties of defining a straight line in the digital sense). As we look at local segments of the line it retains its linearity. In contrast, at line intersections connected sets of pixels do not necessarily retain linearity. Thus, local linearity can be used as a filter to weed out candidate linear pixels from thosc that are part of linear features.

As a measure of linearity we choose the error in a least-squares fit to a line. The procedure used to identify locally linear sets of pixels is as follows for each edge pixel p during a raster scan:

- Identify the locally connected set of edge pixels of like type.
- Suppress p if the number of locally connected neighbors is below a threshold.
- Suppress p if the error of the least squares fit exceeds a threshold.

The effect of this local filter is to suppress noisy edge responses (i.e., edge pixels contained in groups of small nonlinear sets), and to disconnect edge pixels where long linear sets meet at corners. This is illustrated in Figure 5.2.1-1 for a synthetic test image containing both linear and smooth non-linear edges.

5.2.2. Global Analysis of Linearity

It is possible that pixels which are not suppressed by the local linearity filter belong to curves, since curves of large curvature appear linear when viewed locally. The next step is to identify connected sets of pixels and determine if the set is large enough, if the set is linear, and if linear what is the equation of the line. This is easily done by applying a connected components algorithm to remaining edge pixels of like type.

It is necessary, however, to apply a gap filling and line merging algorithm to the resulting lines. The criteria for merging are: (1) is the distance between endpoints small, and (2) do the pixels in the two components fit the same line with little error? Endpoints and parameters in the component files are updated as blobs are merged, and the gap filling pixels are added to the edge map image. Figure 5.2.2-1 shows the final output of the gap-filling stage.

5.3. Second Order Surface Detection

Our first attempt at second order surface detection was an extension of the technique used for planar surface extraction. That is, the relation SIMILAR defined on surface normals of adjacent pixels was used to find smooth surfaces, and then a second order surface fitting was performed on the resulting components. This method was found to be too sensitive to the threshold used in the definition of the SIMILAR relation.

We found a surface growing process to work much better. First, fairly large $\nabla^2 G$ and $\nabla^3 G$ operators are applied to the image to identify image regions which are slowly varying. Within these slowly varying regions a seed is planted, and a second order surface is fit to its neighborhood, which is considered to be a surface patch. Then the following steps are applied repeatedly until the surface patch is not further changed:

1) Points just outside the boundary of the surface patch are added to the surface patch if the distance from the point to the surface is small enough.

2) A new second order surface fit is performed for the surface patch.

3) Points on the boundary of the surface patch are removed from the surface patch if the distance from the point to the surface is too large.

After this process has been applied to enough smooth regions to completely cover the image, each second order fit of each resulting surface patch is classified according to the type of second order surface it determines. We have successfully tested this technique on objects containing sections of cylinders, spheres, elliptic cylinders, ellipsoids, and hyperbolas.

6. TOPOLOGICAL ANALYSIS

The topological analysis module is an algorithm for determining the full set of topological relations existing among the various geometric elements discovered in the geometric processes. The basis of the process is an image growing process which is used to determine two dimensional topology within the image as a guide to three-dimensional topology. Thus this process operates on a combination of processed imagery (the surface and edge map images) and the property list output of the detection processes.

The discussion below applies to polyhedral objects. The more general problem for objects containing the second order surfaces detected above is still under investigation.

One input to the growing process consists of the planar surface map image, in which a pixel with a label (color) of zero indicates the pixel was not selected as a part of any planar surface by the planar surface detections algorithm, and a pixel with a nonzero label indicates the pixel is part of a planar surface component associated with that label.

The other input image is the edge map image, in which pixels have zero or nonzero labels indicating the pixel is a nonedge or edge point, respectively, and, if it is an edge point, the label of the edge.

Consider the union of these two map images. In the union image a pixel will either have a surface region label, an edge region label, or a label of zero indicating the pixel is neither an edge nor a surface pixel. This situation is illustrated in Fig. 6-1.

Now consider applying a growing process (i.e., each region is enlarged by changing non-region pixels to region pixel if they are adjacent to a region border) to the planar surface regions, but with the following conditions:

- Planar surface regions may not grow across edge regions;

- Planar surface regions may not touch each other.

When the growing procedure has been applied iteratively until no more growing is possible, the following occurs: A pixel will have either a surface label, an edge label, or will have a label of zero and will be adjacent to two or more surfaces. This situation is illustrated in Fig. 6-2.

Two surfaces can only meet at edges or at vertices, so the pixels with labels of zero must indicate the topological existence of vertices. Thus, the growing process is followed by a connected components analysis applied to all the pixels that are neither edge nor surface pixels. At the completion of this process every pixel is labeled uniquely as being part of a surface, an edge, or a vertex.

Once the growing is complete and the vertices have been labeled it is easy to recover the topological relations among the various region types within the image:

- If a pixel has a surface label S and one of its neighbors has an edge label E, then we can form the relations ADJACENT(S,E) and ADJACENT(E,S);

- If a pixel has an edge label E and one of its neighbors has a vertex label V, then we can form the relations ADJACENT(E,V) and ADJACENT(V,E).

These relations refer to adjacency within the image and not to connectivity in three-space. For example, if a surface S1 occludes another surface S2 at an occluding edge E, E is ADJACENT to S1 and to S2 but is not connected in three-space.

Because of this difference between image adjacency and three-space connectedness a number of consistency checks are applied to the adjacency relations in order to compute the three-dimensional connectedness relations. Thus, these consistency checks may remove or modify ADJACENCY relations in order to create the three-dimensional connectedness relations.

As a final step in the topological routines, the set of edges forming the boundary of a surface are ordered into a counter-clockwise traversal of the boundary. At the same time the vertices which connect the edges of the boundary are also placed into counter-clockwise order.

After the topological processing is complete the topological existence of vertices has been determined, but not their geometric locations. Three-dimensional locations of vertices are computed from the combined topology and geometry information for surfaces and edges. For each vertex, the topological information is used to acquire the relevant geometric information by determining which surfaces and edges combine to form the vertex. Based on the extent to which the geometry of connected surfaces and edges is known, one of three cases applies:

- If a vertex has three nonoccluding edges then three visible surfaces intersect at the vertex; the computed intersection of the three surfaces is the location of the vertex.

- If a vertex has three edges but only one is nonoccluding then only the two visible surfaces adjacent to the nonoccluding edge intersect the edge; the intersection of these two surfaces with the surface defined by the focal point of the sensor and the end-points of either occluding edge is the location of the vertex.

- If a vertex has only two visible edges then only the visible surface between these two edges intersects the vertex; this surface is intersected with the two planes defined by the focal point of the sensor and the end-points of each edge.

Surfaces are used in the above computations even though in the first two cases the edge data alone could be used to locate the edge. There are two reasons for this. First, is that because there are more data points contributing to the calculation of surface geometry than to edge geometry noise in the data is averaged to a greater extent. Second, because edges occur at discontinuities of the derivatives of range values they are are less reliable.

At the completion of this process the topological connectedness information form arcs and the geometric elements and their parameters form nodes of a graph-like structure called the scene-model. This scene-model is then given to the object recognition modules for subsequent processing.

7. SYMBOLIC PROCESSING: OBJECT RECOGNITION

The approach of the object recognition system is to transform the information produced by the previous processes, to construct a relational representation of this information, and then obtain the best match or matches between the image-model and object-models in the object-model data-base. The process which does this is a goal directed rule-based system. This approach was chosen because it is well suited for pattern matching and permits reasonably fast results.

7.1. System Components

A fundamental knowledge of the planar, three-dimensional world is first encoded. Domain and application-independent knowledge of topological and geometrical facts is constructed as a set of facts, called *assertions*, in a "theory" of the MRS representation language. This theory, labeled TOPOLOGY, describes the manner in which surfaces, edges, and vertices may be found. The TOPOLOGY theory stems from the work of Clowes[15] .

The TOPOLOGY theory also contains information about incomplete junctions and how to enter missing data. This is useful because junction components for any object may be missing due to object self-occlusion and the limitations of the geometric feature extraction programs. These limitations, however, will not necessarily result in an unsuccessful attempt at recognition. Instead, the TOPOLOGY theory will suggest to a higher level theory that certain edge data may be missing. If the recognition process can be performed in a logically consistent manner using the inferred missing information then the system will assume this, complete the match and send a message stating that is suspects an edge to be present at some location. This information could be used to guide a second pass of the geometric feature extraction processes.

Knowledge specific to the inspection task is represented in a separate theory called MODELS that describes the *a priori* shape information of the objects relevant to the inspection task. The assertions describing the object *one-step* are:

```
(if (and (tri+-+* #p1 #p2 #p7 #p6)
         (tri+-+* #p7 #p8 #p1 #p12)
         (tri+++* #p2 #p1 #p3 #p8)
         (tri+++* #p3 #p2 #p4 #p9)
         (tri+++* #p4 #p3 #p5 #p10)
         (tri+++* #p5 #p4 #p6 #p11)
         (tri+++* #p6 #p1 #p5 #p12)
         (tri+++* #p7 #p1 #p8 #p12)
         (tri+++* #p8 #p2 #p7 #p9)
         (tri+++* #p9 #p2 #p3 #p8)
         (tri+++* #p10 #p4 #p9 #p11)
         (tri+++* #p11 #p5 #p10 #p12)
         (tri+++* #p12 #p6 #p7 #p11) )
     (one_step #p1 #p2 #p3 #p4 #p5 #p6 #p7 #p8 #p9
               #p11 #p12
     )
)
```

The final set of facts is contained in the theory DATA, which describes the scene-models in a relational representation. That is, the graph-like structure produced by the preceding geometric feature extraction processes and the topological analysis processes are transformed into assertions describing the information derived from the image. Examples of these assertions, from the *one-step*, are:

```
(convex v1 v3 1)      {* The last field is length *}
(convex v2 v3 1)
(concave v1 v7 3)
(occluding v3 v4 2)
.
.
.
(angle v6 v1 v2 -90)
(angle v1 v2 v3 +90)
.
.
.
(in_plane v1 s1)
(in_plane v2 s1)
```

The problem for the symbolic processing module then is to determine which object in the MODELS theory is represented in the DATA theory. Another way of describing this problem is to say that the task of the symbolic processing module is to prove one of the theorems contained in the MODELS theory given the facts contained in the DATA theory.

In order to constrain this search the matching process is divided into the following three stages:

1) Inhibition of MODELS for which there is contrary evidence;

2) Reinforcement of uninhibited MODELS using evidential rules;

3) Fully instantiating one of the candidate MODELS to obtain recognition.

The first step removes from further consideration as quickly as possible those MODELS which are obviously incorrect. Examples of these rules are: "If the DATA contains a concave edge then remove from further consideration all convex objects", "If the DATA contains a surface with five sides then remove from further consideration all convex objects without a five sided surface", and "If the DATA contains a surface with area A then remove from further consideration all surfaces with no side of area B, $A-\varepsilon \leq B \leq A+\varepsilon$, for some small ε ."

The second step uses evidential rules to constrain or order by priority the remaining MODELS. Generally these rules should seek some unique feature within DATA that distinguishes a certain MODEL from other MODELS. As with the inhibition rules, these evidentiary rules may look for uniqueness in concave or convex edges, number of sides to a surface, existence of certain non-planar surfaces, surface area, or combinations of these to constrain objects at higher levels.

It is important to note that these first two steps serve only to speed the recognition processing. If there are only a small number of rules for each step, or if they are not very efficient at constraining the search, the object will still be recognized in the third step, but this third step will take longer. Conversely, if the recognition processing is taking too long it can improved by adding rules to these steps. Thus, a simple system will recognize any object, but the system can be tailored to recognize specific objects quickly for a certain application.

The third step is to instantiate one or more of the MODELS. We use the term "instantiate" rather than "recognize" because DATA may appear as one or more MODELS depending on viewpoint. For example, a cube and the one_step shown earlier may both appear as cubes if the one_step is viewed from behind or below.

The system is currently capable of performing scene analysis on simple polyhedral objects, on objects containing holes, and on objects containing second order surfaces bounded by edges. We are working on expanding the set of geometries which can be recognized by the system.

8. FROM WHERE DO MODELS COME?

The operation of the range image processing system is application independent up to the point where MODELS are used. These MODELS describe the particular objects which form the domain of the inspection process. Thus, in order to use the system these MODEL descriptions must be added to the data base in an application dependent fashion.

There are two possible sources for the data contained in MODELS. The first, and most obvious, is to use CAD/CAM data describing the objects. Most CAD/CAM renderings of objects contain all the geometric and topologic information necessary to encode the object as a MODEL.

Frequently, however, for any number of good reasons no CAD/CAM rendering of an object exists. The second approach to obtaining this data is to use the inspection system itself to generate a CAD/CAM rendering of the object. That is, the first steps of recovering geometry and determining topology produce sufficient information to produce the representation required for a MODEL.

When used in this fashion the resulting CAD/CAM model represents only the visible side of the object. Another process has been developed wherein given such a partial model, vantage points for viewing hidden portions of the object can be computed. The robotic manipulator can then reposition the sensor to obtain a model of the previously hidden side of the object, and the two models are combined. This process is repeated until no portions of the object are omitted (with the exception of cavities or deep holes). An advantage of using this approach is that the object is modeled in terms of the geometrics and topologies recognizable by the inspection system.

9. MENSURATION

The laser range imaging sensor can be used directly for the task of performing high-precision measurements. When these tasks are to be performed the object recognition and location processes are used first to determine the best vantage point for making the measurement. A robotic manipulator then is used to position the sensor for making the measurement.

Different mensuration algorithms are available according to the type of measurement to be made. The current set of measurement types are: thickness, height, and flange curvature. These algorithms are described briefly below.

- Rib Or Flange Height-- Two patches of range data are collected as shown in Fig. 7-1. The distances from each point in the first patch to the surface of the second patch, measured along the direction of the surface normal, are computed. These height may be either averaged (if the top of the flange is flat and parallel to the surface of patch 2), or the maximum value may be chosen.

- Rib Or Flange Thickness-- Two patches of range data are collected as shown in Fig. 7-2. Edges of the rib are located in patch 1, and the surface of patch 2 is computed. The width of the rib is then computed as the distance between the two edges in the direction defined by the cross product of the surface normal and the edges.

•Circular Fillet Radius-- Three patches of range data are collected as shown in Fig. 7-3. The direction of the fillet spine is defined as the cross product of the normal vectors of the surfaces of patches 1 and 2. The fillet radius is measured from the best least squares fit of the data of patch 3 to a cylinder with the spine as determined above.

10. CONCLUSION

We have described a computer vision system for automated dimensional inspection. Because the system is organized into application independent processes and application dependent data the system is easily extendible to other inspection tasks. In fact, because the essence of the system is object recognition and location, which is prerequisite for many autonomous tasks besides inspection, the system is directly usable for a variety of applications. Applications for which this system is not applicable are generally those for which range data is not suitable, such as for character recognition, inspecting for paint or other coatings, or inspecting for texture/roughness.

REFERENCES

[1] S.W. Holland, L. Rossol, and M.R. Ward, "CONSIGHT-1: A Vision-Controlled Robot System for Transferring Parts from Belt Conveyors." *Computer Vision and Sensor-Based Robots*, ed. G. G. Dodd and L. Rossol, Plenum Press, New York, 1979.

[2] W.B.Gevarter, "An Overview of Computer Vision." National Bureau of Standards Report NBSIR 82-2582, September, 1982.

[3] B.K.P. Horn, "Obtaining Shape From Shading Information." *The psychology of Computer Vision*. ed. P. H. Winston, McGraw Hill, New York, 1975.

[4] T.O. Binford, "Inferring Surfaces from Images." *Artificial Intelligence* 17, 1981, pp205-244.

[5] J.R. Kender, "Shape from Texture: A Computational Paradigm." *Proceedings of the Darpa Image Understanding* Workshop," Palo Alto, CA, 1979.

[6] A. P. Witkin, "Shape from Contour", Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, 1980.

[7] K Ikeuchi, and B.K.P. Horn, "Numerical Shape from Shading and Occluding Boundaries." *Artificial Intelligence*,

[8] R.A. Brooks, "Symbolic Reasoning Among 3-D Models and 2-D Images," *Artificial Intelligence*17,

[9] R.A. Brooks, R. Greiner, and T.O. Binford, "The ACRONYM Model-Based Vision System", *Proceedings of the 6th International Joint* Conference on Artificial Intelligence", Tokyo, 1979,pp 105-113.

[10] D. Marr, "Representing Visual Information", *Computer Vision Systems*, M. Riseman, Academic Press, New York, 1978.

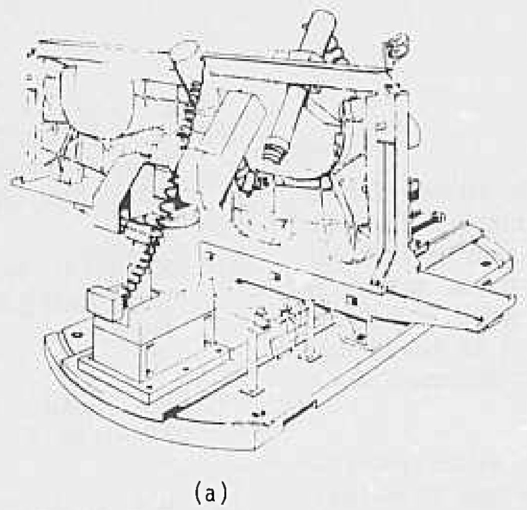
[11] M. A. Wesley, T Lozana-Perez, L.I. Lieberman, M.A. Lavin, and D.D. Grossman, "A Geometric Modeling System for Automated Mechanical Assembly", *IBM Journal of Research and Development*24(1), 1980, pp 64-74.

[12] D.M. Zuk and M.L. Dell'eva, "Three-Dimensional Vision System for the Adaptive Suspension Vehicle," Environmental Research Institute of Michigan technical report, Defense Advanced Research Projects Agency report number 170400-3-F, December 1982.

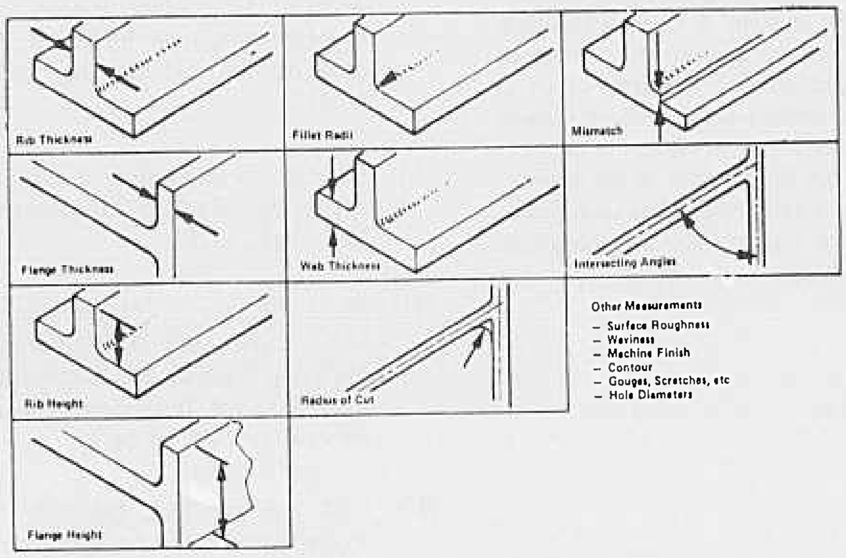
[13] "Phase 1 Intelligent Task Automation Final Report," Martin Marietta Aerospace, Air Force Wright Aeronautic Laboratory/Defense Advanced Research Projects Agency contract number F33615-82-C-5139, Feb, 1985.

[14] D. Marr, and E.C. Hildreth, "Theory of Edge Detection," *Proceedings of the Royal Society of London*, Series B, 207, 1980, pp 187-217.

[15] M.B. Clowes, "On Seeing Things," *Artificial Intelligence*, vol. 2, 1971, pp 79-116.



(a)



(b)

Figure 3.1-1. (a) Artists rendition of robotic work area showing F-15 bulkhead, (b) Measurements to be performed.

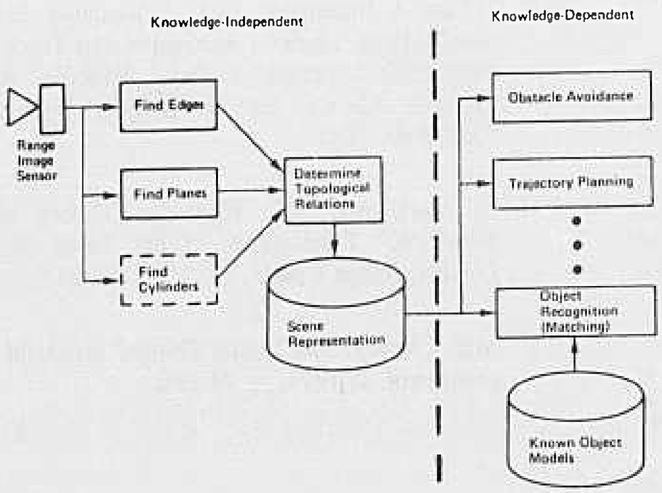


Figure 4.1
Range image processing system functional block diagram.

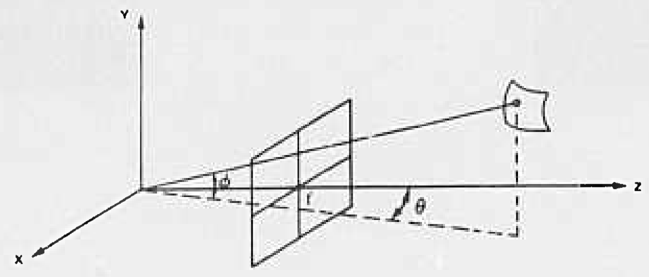
-- simple perspective with image plane at $z = f$

-- pixel P(row,col) is located at $(R_p, \theta_p, \emptyset_p)$

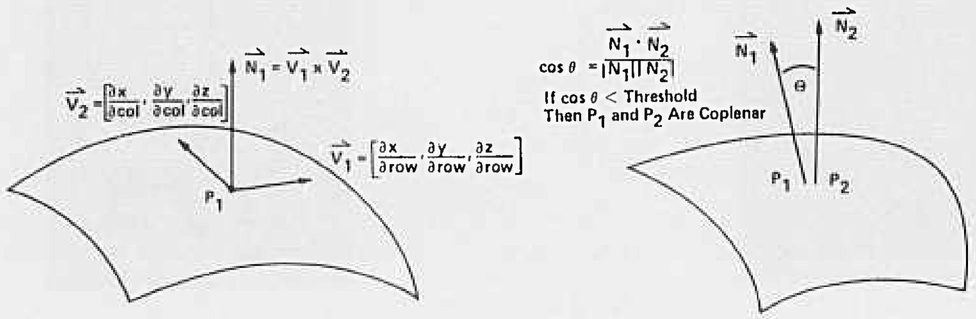
$$R_p = (row^2 + col^2 + f^2)^{1/2}$$
$$\theta_p = \sin^{-1}\left(\frac{col}{R_p}\right)$$
$$\emptyset_p = \sin^{-1}\left(\frac{row}{R_p \cos\theta_p}\right)$$

-- transform to cartesian coordinates

$$x(row,col) = x(\theta_p, \emptyset_p) = R(\theta_p, \emptyset_p) \sin\theta_p \cos\emptyset_{1/2}$$
$$y(row,col) = y(\theta_p, \emptyset_p) = R(\theta_p, \emptyset_p) \sin\emptyset_p$$
$$z(row,col) = z(\theta_p, \emptyset_p) = R(\theta_p, \emptyset_p) \cos\theta_p \cos\emptyset_p$$



(a)


$\vec{N_1} = \vec{V_1} \times \vec{V_2}$
$\vec{V_2} = \left[\frac{\partial x}{\partial col}, \frac{\partial y}{\partial col}, \frac{\partial z}{\partial col}\right]$
$\vec{V_1} = \left[\frac{\partial x}{\partial row}, \frac{\partial y}{\partial row}, \frac{\partial z}{\partial row}\right]$
P1
$\cos\theta = \frac{\vec{N_1} \cdot \vec{N_2}}{|N_1||N_2|}$
If cos θ < Threshold
Then P1 and P2 Are Coplanar
$\vec{N_1}$
$\vec{N_2}$
θ
P1
P2

(b)

Figure 5.1-1. (a) Range sensor geometry model
(b) Planor surface detection computation.

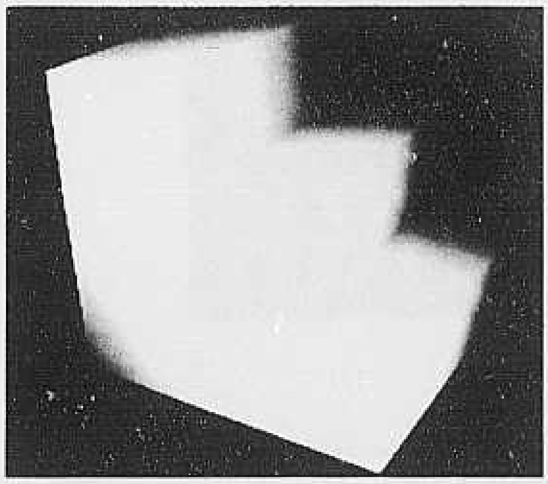
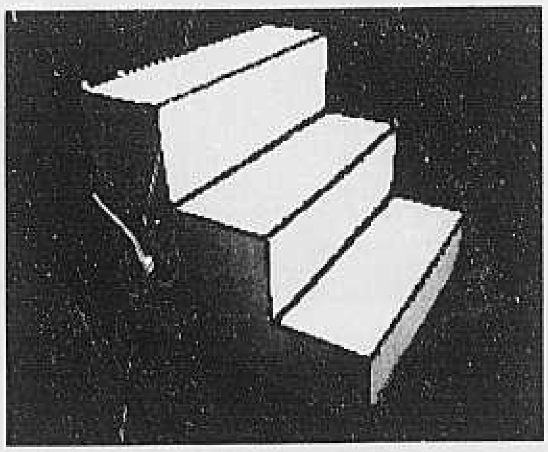



Figure 5.1-2 (a) Original range image (b) Planor surface map image.

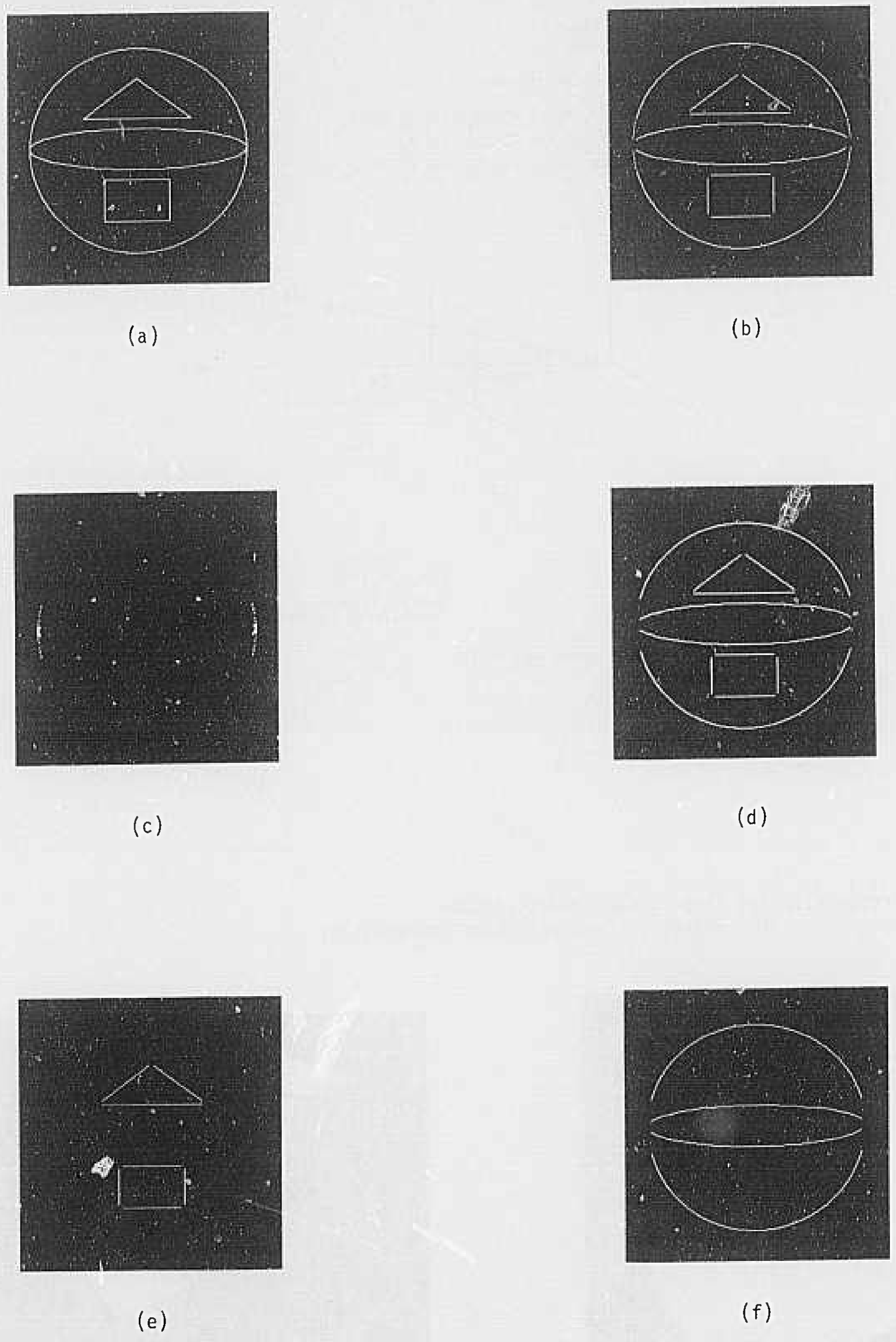
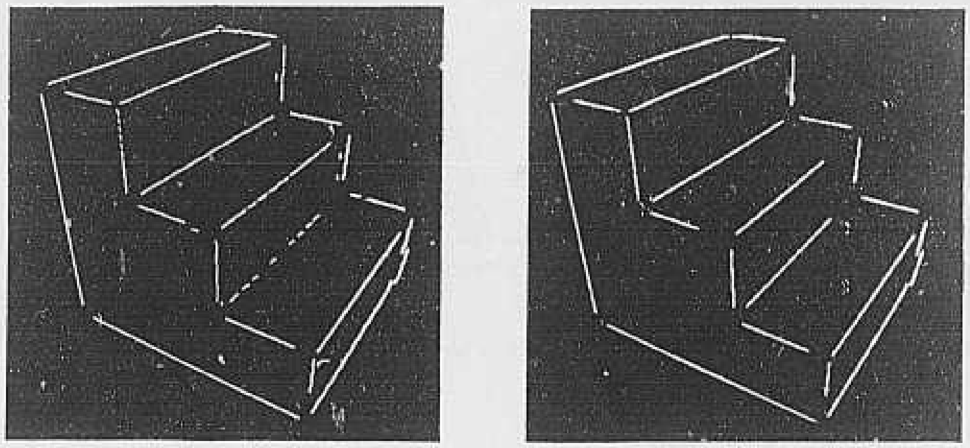


Figure 5.2.1-1. The edge point aggregation process (a) Synthetic Line drawing with curved and linear features, (b) Locally Linear Features, (c) Locally non-linear features, (d) Remaining connected components, (e) Globally linear features, (f) Globally curved features.



(a) (b)

Figure 5.2.2-1. (a) Before, and (b) after gap filling process.

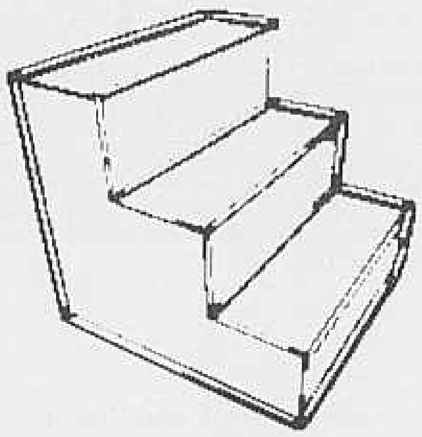


Figure 6-1. Union of the Edge map and the Surface map.

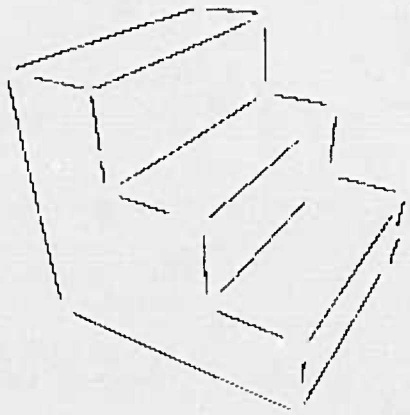
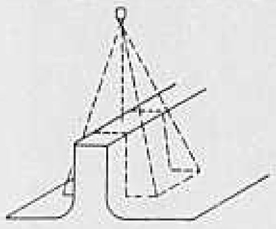
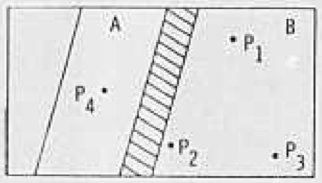


Figure 6-2. After surface growing vertices are detected.

Rib Height Measurement Algorithm





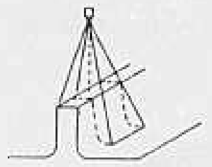
$P_1 = (x_1, y_1, z_1)$, $P_2 = (x_2, y_2, z_2)$, $P_3 = (x_3, y_3, z_3)$ are used to determine the equation of the web plane via:

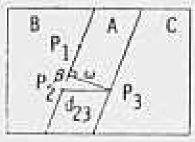
$$Ax + By + Cz + D = 0$$

Distance from a point $P_4 = (x_4, y_4, z_4)$ to this plane along a line perpendicular to the plane is:

$$d = \frac{|Ax_4 + By_4 + Cz_4 + D|}{(A^2 + B^2 + C^2)^{1/2}}$$

Figure 7-1. Rib Height measurement.





$$\beta = \text{arc cos}\left[\frac{(x_1 - x_2)(x_2 - x_3) + (y_1 - y_2)(y_2 - y_3) + (z_1 - z_2)(z_2 - z_3)}{d_{12}\, d_{23}}\right]$$

$$d_{12} = \left[(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2\right]^{1/2}$$

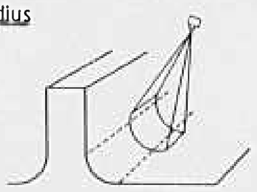
$$d_{23} = \left[(x_2 - x_3)^2 + (y_2 - y_3)^2 + (z_2 - z_3)^2\right]^{1/2}$$

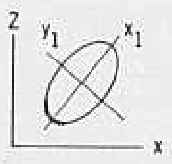
$$\omega = d_{23} \sin$$

Figure 7-2. Rib Thickness measurement.

Radius Measurement Algorithm

Fillet Radius





① Points on An Image Plan $y = y_0$ Lie on An Ellipse

② $Ax_1^2 + Cy_1^2 + Dx_1 + Ey_1 + F = 0$

Describes the ellipse in an x_1, y_1 coordinate system resulting from a rotation of the x, y coordinate system by an angle α.

③ This may be rewritten as:

$$\frac{\left(x_1 + \frac{D}{2A}\right)^2}{\xi / A} + \frac{\left(y_1 + \frac{E}{2C}\right)^2}{\xi / C} = 1$$

where: $\xi = \left(-F + \frac{D^2}{4A} + \frac{E^2}{4C}\right)$

④ The radius of the cylinder (fillet) is given by

$R_c = \min [\sqrt{\xi / A}, \sqrt{\xi / C}]$.

Figure 7-3. Fillet radius measurement.

Vision and Navigation for the Carnegie Mellon Navlab

Charles Thorpe
Steven Shafer
Takeo Kanade
and the members of
the Strategic Computing Vision Lab

1. Introduction

Robotics is where Artificial Intelligence meets the real world. AI deals with symbols, rules, and abstractions, reasoning about concepts and relationships. The real world, in contrast, is tangible, full of exceptions to the rules, and often stubbornly difficult to reduce to logical expressions. Robots must span that gap. They live in the real world, and must sense, move, and manipulate real objects. Yet to be intelligent, they must also reason symbolically. The gap is especially pronounced in the case of outdoor mobile robots. The outdoors is constantly changing, due to wind in trees, changing sun positions, even due to a robot's own tracks from previous runs. And mobility means that a robot is always encountering new and unexpected events. So static models or preloaded maps are inadequate to represent the robot's world.

The tools a robot uses to bridge the chasm between the external world and its internal representation include sensors, image understanding to interpret sensed data, geometrical reasoning, and a concept of time and of the vehicle's motion over time. We are studying those issues by building a mobile robot, the CMU Navlab, and giving it methods of understanding the world. The Navlab has perception routines for understanding color video images and for interpreting range data. CODGER, our "whiteboard", which was developed for the Navlab and its smaller cousin the Terregator, handles much of the modeling of time and geometry. Our architecture coordinates control and information flow between the high-level symbolic processes running on general purpose computers, and the lower-level control running on dedicated real-time hardware. The system built from these tools is now capable of driving the Navlab along narrow asphalt paths near campus while avoiding trees and pausing for joggers that get in its way.

Navlab construction is described elsewhere [11]. Briefly, it is a van converted into a robot, capable of being driven conventionally or under computer control. The Navlab is self contained, carrying its own sensors, processors, power, and even researchers.

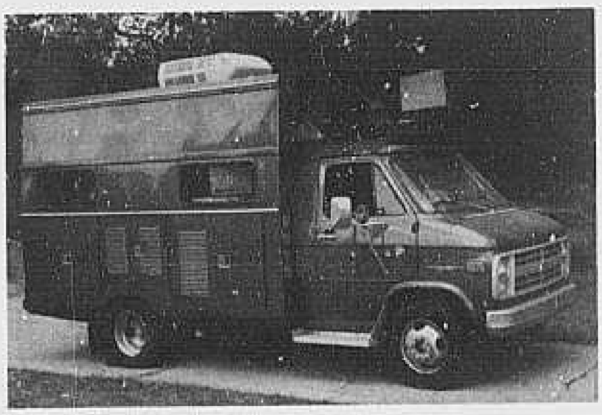


Figure 1. The Navlab.

This report describes the software we have built for the Navlab over the past year: color vision, for finding and following roads [12]; 3D perception, for obstacle avoidance [4]; and the CODGER whiteboard [10].

2. Color Vision

The Navlab uses color vision, specifically multi-class adaptive color classification, to find and follow roads. Image points are classified into "road" or "non-road" on the basis of their color. Since the road is not a uniform color, color classification has to have more than one road model, or class, and more than one non-road class. And since conditions change from time to time and from place to place over the test course, the colors have to adapt. Once the image is classified, the road is found with an area-based voting technique that finds the most likely location for the road in the image.

2.1 Vision Principles for the Real World

We based the development of our vision on the following principles:

Assume variation and change. On sunny days, there are shadowed areas, sunlit areas, and patches with dappled sunlight. On rainy days, there are dry patches and wet patches. Some days, there are wet and dry and sunny and shadowed all in the same image. The road has clean spots and other places covered with leaves or with drips of our own hydraulic fluid. And as the sun goes behind a cloud or as the vehicle turns, lighting conditions change.

This means that we need to have more than one road color class and more than one non-road class, that those classes need to adapt to changing conditions, and that we need to process images frequently so the change from one image to the next will be moderate.

Use few geometric parameters. A complete description of road's shape in an image can be quite complex. The road can bend gently or turn abruptly, can vary in width, and can go up or down hill. However, the more parameters there are, the greater the chance of error in finding those parameters. Small misclassifications in an image could give rise to fairly large errors in perceived road geometry. Furthermore, if all the road parameters can vary, there are ambiguous interpretations: does the road actually rise, or does it instead get wider as it goes? We chose to describe the road with only two free parameters: its orientation and its distance from the vehicle. Road width is fixed, we assume a flat world, and we decree that the road is straight. While none of those assumptions are true over a long stretch of the road, they are nearly true within any one image. And the errors in road position that come from those oversimplifications are balanced by less chance of bad interpretations. If there are a few pixels incorrectly classified as road, the worst our method will do is to find a slightly incorrect road. A method that tries to fit more parameters, on the other hand, may come up with the perfect interpretation of part of the road, but could find an abrupt turn or sudden slope near the bad pixels.

Work in the image. The road can either be found by projecting the road shape into the image and searching in image coordinates, or by projecting the image onto the ground and searching in world coordinates. The problem with the latter approach comes in projecting the image onto an evenly spaced grid in the world. The points on the world grid close to the vehicle correspond to a big area in the lower part of the image; points farther away may correspond to one or a few pixels near the top. So either there has to be a complex weighting scheme, or some image pixels (those at the top, that project to distant world points) will have more weight than other (lower) points. A few noisy pixels can have a big or a small effect, depending on where in the image they lie. On the other hand, working directly in the image makes it much easier to weight all pixels evenly. We can directly search for the road shape that has the greatest number of road pixels and the least non-road pixels. Moreover, projecting a road shape is much more efficient than projecting all the image pixels.

Calibrate directly. A complete description of a camera has to include its position and orientation in space; its focal length and aspect ration; lens effects such as fish-eye distortion; and non-linearities in the optics or sensor. The general calibration problem of trying to measure each of these variables is very difficult. It is much easier, and more accurate, to calibrate the whole system rather than trying to tease apart the individual parameters. The easiest method is to take a picture of a known object and build a lookup table that relates each world point to an image pixel and vice versa. Projecting and deprojecting is simply done by table lookup, or table lookup for close by values with simple interpolations. Building such a table is straightforward, it provides good accuracy, and there are no instabilities in the calculations.

Use outside constraints. Even without a map of our test course or an expensive inertial navigation system, we know approximately where the road should be based on the previous image and on vehicle motion. The whiteboard can predict where the road should appear if the road were straight and the vehicle navigation were perfect. Adding a suitable margin for curved roads and sloppy navigation still gives useful limits on where in the image to look for the road.

Test. We tried to run our VCR every day we took the vehicle out, to collect images under as many conditions as possible. We had sunny days, cloudy days, rainy days, leaves on trees, leaves turning color, leaves falling, early morning, noon, after dusk, even a partial solar eclipse. Strategies that worked well on one set of images did not always work on the others. Our philosophy was to collect the toughest images from those sets. We ran our best algorithms and printed the classification results, changed parameters or algorithms, reran the data set, and compared results. This gave us the best chance of being methodical and of not introducing new bugs as we went. When the image processing worked to our satisfaction, we ran simulations in the lab that included the whiteboard, range processing, path planning, and a vehicle simulator, with vision processing stored images and interacting with the rest of the system. When the simulations worked in the lab, we moved them to the vehicle. Only after the simulations worked on the vehicle's computers, and we were sure that all necessary software was on the van, did we go into the field for real tests. Even then, everything didn't work, but there were many fewer bugs than there would have been without the simulations and tests.

2.2 Road Following Algorithm

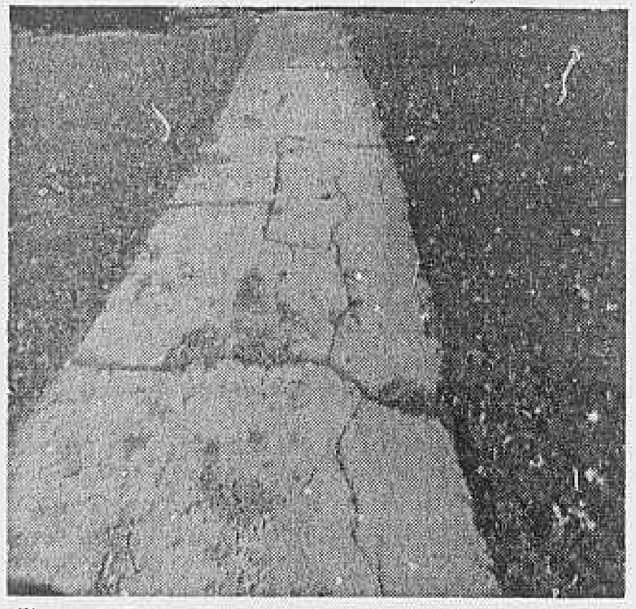


Figure 2. Original image.

We followed those principles in building and tuning adaptive color classification for following roads. Our algorithm involves three stages:

1. Classify each pixel.

2. Use the results of classification to vote for the best-fit road position.

3. Collect new color statistics based on the detected road and nonroad regions.

Pixel classification is done by standard pattern classification. Each class is represented by the means, variances, and covariances of red, green, and blue values, and by its a priori likelihood based on expected fraction of pixels in that class. For each pixel, calculating the class to which it most likely belongs involves finding how far the pixel's values lie from the mean of each class, where distance is measured in standard deviations of that class.

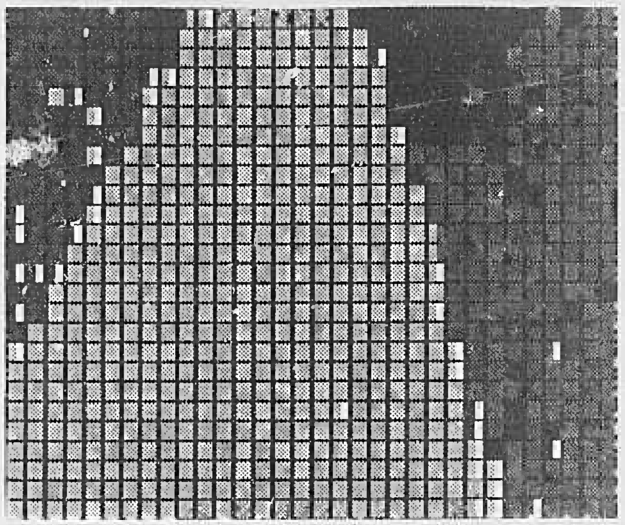


Figure 3. Segmented image. Color and texture cues are used to label points below the horizon into two road and two offroad classes.

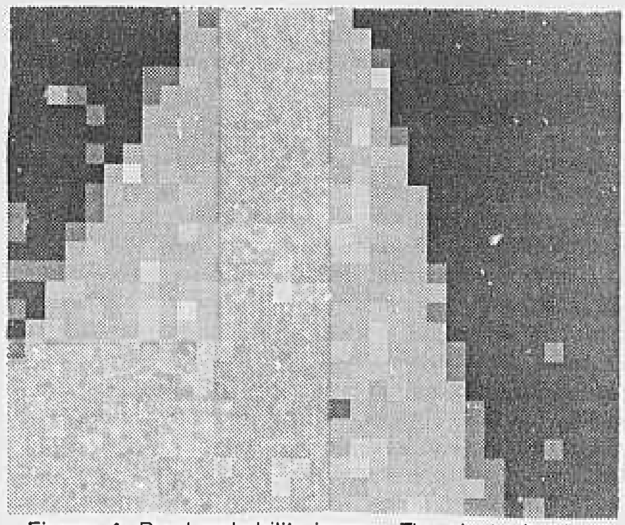


Figure 4. Road probability image. The pixels that best match typical road colors are displayed brightest.

Once each point has been classified, we have to find the most likely location of the road. We assume the road is locally straight, and can be described by two parameters:

1. The image column of the road's "vanishing point", where the road intercepts the horizon. This gives the road's direction relative to the vehicle.

2. The orientation of the road in the image, which gives how far the vehicle is to the right or left of the centerline.

We set up a two dimensional parameter space, with intercept as one dimension and orientation as the other. Each point classified as road votes for all road intercept / orientation combinations to which it could belong. The orientation / intercept pair that receives the most votes is the one that contains the most road points, and is reported as the road.

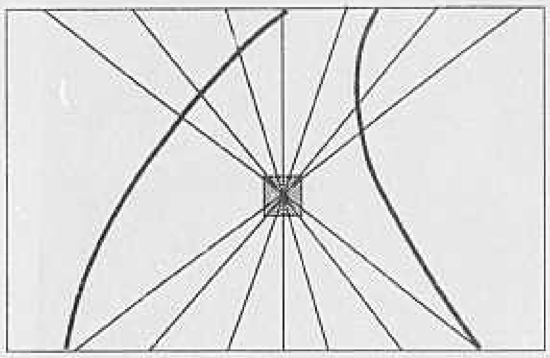


Figure 5. A point classified as road could be a part of roads with different orientations and vanishing points.

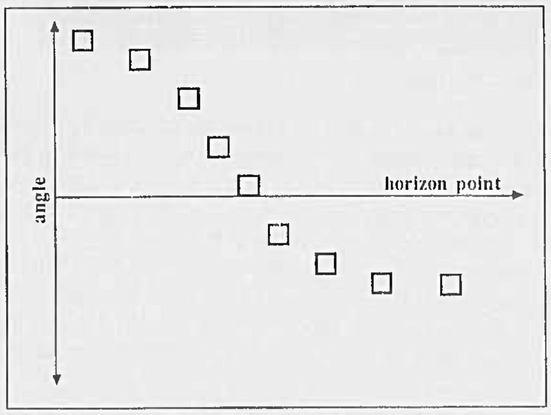


Figure 6. The point from figure 5 would vote for these orientation / intercept values in the parameter space.

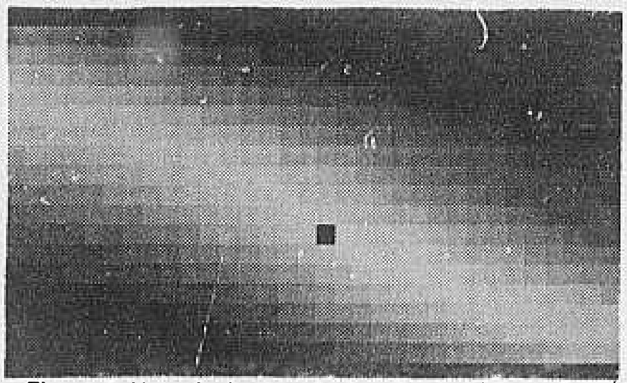


Figure 7. Votes for best road orientation and intercept, and point with most votes.

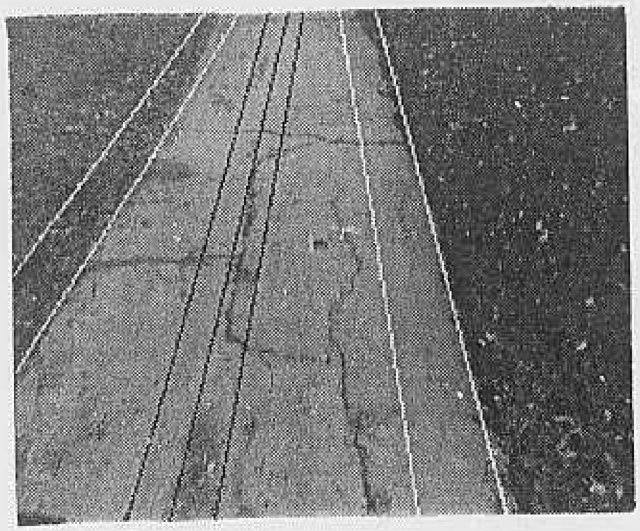


Figure 8. Detected road.

Once the road has been found in an image, the color statistics are recalculated for each class. The updated color statistics will gradually change as the vehicle moves into a different color road, or as lighting conditions change, or as the color of the surrounding grass varies. As long as the processing time per image is low enough that there is a large overlap between images, the statistics adapt as the vehicle moves. The road is picked out by hand in the first image. Thereafter, the process is automatic, using the segmentation from each image to calculate color statistics for the next.

There are several variations on this basic theme. One variation is to smooth the images first. This throws out outliers, and tightens the road and nonroad clusters. Another is to have more than one class for road and for non-road, for instance one for wet road and one for dry, or one for shadows and one for sun. If we knew where the wet road and dry road patches were, we could collect separate color statistics for those two classes. Since we don't know where those areas are, we collect new statistics by starting with one class for the top part of the road image and one for the bottom. Using color statistics from those regions, we reclassify the road pixels in the same image, and recalculate statistics based on this reclassification. We loop through the classify - update cycle several times for each image. As long as the first two point sets have different distributions, the resulting classes will move towards separate, tighter clusters.

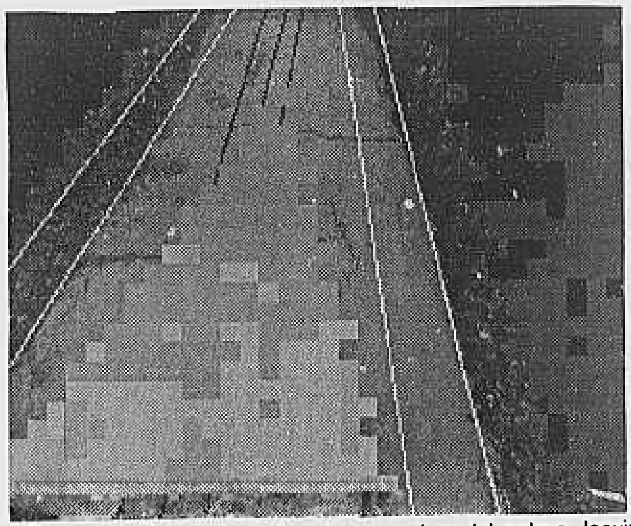


Figure 9. Updating road and nonroad model colors, leaving a safety zone around the detected road region.

Other variations change the voting for best road. Besides adding votes for road pixels, we subtract votes for non-road points. And votes are weighted according to how well the point matches road or nonroad classes.

There are other clues in an image besides color, for instance visual texture. Roads tend to be smooth, with less high-frequency variation than grass or leaves. Our first texture algorithm ran a Robert's gradient operator, thresholded the result, and counted the number of pixels above threshold in each 16 by 16 block. This method fails to deal adequately with shadow edges, to which Robert's operator responds strongly, or with the interiors of shadows, in which the absolute values of gradients are small because the pixel values are small. The shadow edge problem can be handled by dividing the high frequency gradient by a lower frequency gradient, obtained by running Robert's operator on a reduced resolution image. Grass and leaf texture will show up only at high resolution, while shadow edges show up at both. Shadow interiors can be normalized by dividing by the average pixel intensity. We calculate a normalized gradient as:

$$\frac{\text{high-frequency gradient}}{\alpha \times \text{low-frequency gradient} + \beta \times \text{average pixel value}}$$

We threshold the normalized gradient and count the number of pixels above threshold in each block.

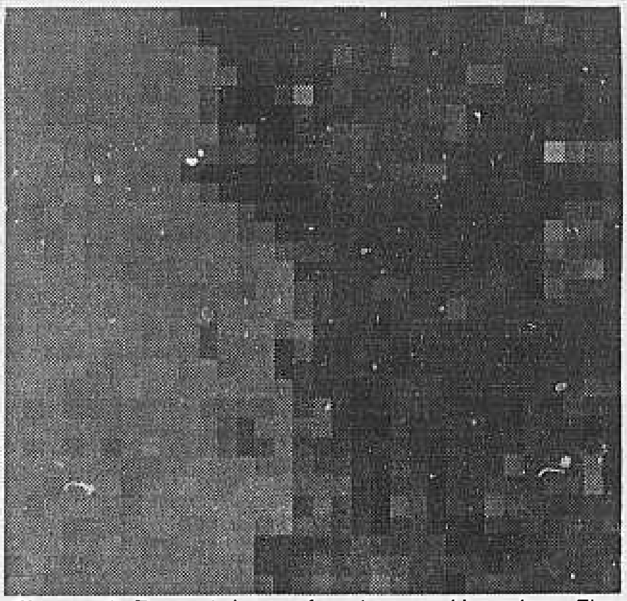


Figure 10. Zoomed picture of road-nonroad boundary. The road (at left) is much less textured than the grass (at right).

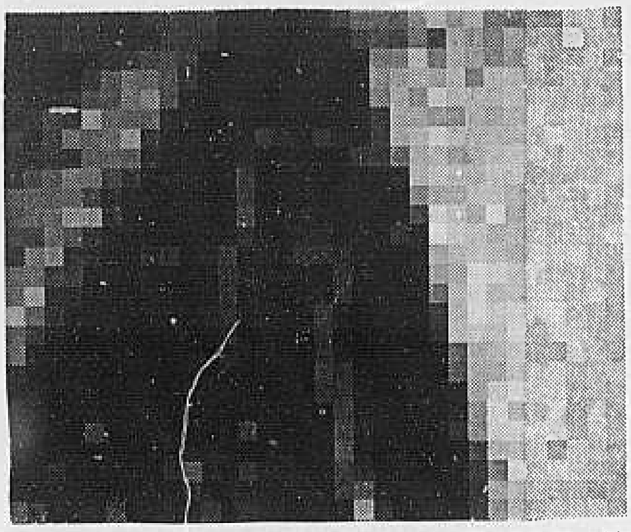


Figure 11. Low resolution texture image. The brighter blocks are image areas with more visual texture.

2.3 Implementation and Results

The best combination we have found is to have two road classes and two non-road classes, and to smooth the image with a 16 by 16 averaging filter. The two road classes handle sunny and shaded road on bright days, or wet and dry patches under overcast conditions. The two non-road classes often converge on trees and grass, or grass and dry leaves. Filtering the images by averaging reduces the effect of cracks in the sidewalk and other scene anomalies, and also smooths out color aberrations from misaligned camera guns and from noise in the digitizers and color splitter.

The texture information could be used as a fourth element in the classification along with red, green, and blue. We found texture used this way to be less reliable than color. Instead, we treat texture as a separate feature with fixed statistics (mean and variance for road and nonroad), used in a post processing stage. We calculate road and nonroad probabilities based on color and separately based on texture. We then combine the two with color weighted much more heavily, but texture discriminating in borderline cases.

This algorithm correctly classifies over 96% of our test images in the lab. When we run on the Navlab, with predictions of where the road should appear, our failure rate is very close to 0. The occasional remaining problems come from having too many leaves on the road, so the color statistics of one road class and of nonroad are very close. Not every image is classified perfectly, but almost all are good enough for navigation. We leave a 50 pixel "safety zone" (about two feet on the ground near the vehicle) along the road border that is not used in updating color statistics, to keep from being drawn off by small mistakes or curving roads.

There is no need for complete geometric calibration. The vision algorithms calculate the road's shape (road width and location of the horizon) from the first training image. We also take two calibration pictures, with a meter stick placed perpendicular to the vehicle, 8 and 12 meters in front. Then, during the run, given the centerline of a detected road in image coordinates, it is easy to get the x position of the road at 8 and 12 meters, and then to calculate the vehicle's position on the road.

This algorithm runs in about 10 seconds per image on a dedicated Sun 3/160, using 480 by 512 images reduced to 30 by 32 pixels. We currently process a new image every 4 meters, which gives about three fourths of an image overlap between images. Ten seconds is fast enough to balance the rest of the system, but is slow enough that clouds can come and go and lighting conditions change between images. We plan to port this algorithm to the Warp, CMU's experimental high-speed processor. We hope to process an image per second, and to use higher resolution.

3. 3D Perception

Our obstacle detection starts with direct range perception using an ERIM scanning laser rangefinder. Our ERIM produces, every half second, an image containing 64 rows by 256 columns of range values. The scanner measures the phase difference between an amplitude-modulated laser and its reflection from a target object, which in turn provides the distance between the target object and the scanner. The scanner produces a dense range image by using two deflecting mirrors, one for the horizontal scan lines and one for vertical motion between scans. The volume scanned is 80 degrees wide and 30 high. The range at each pixel is discretized over 256 levels from zero to 64 feet.

Our range processing begins by smoothing the data and undoing the peculiarities of the ranging geometry. The "ambiguity intervals", where range values wrap around from 255 to 0, are detected and unfolded. Two other undersirable effects are removed by the same algorithm. The first one is the presence of mixed points at the edge of an object. The second is that a measurement from a surface such as water, glass, or glossy pigments is meaningless. In both cases, the resulting points are in regions limited by considerable jumps in range and can therefore be removed by the same algorithm. We then transform the values from angle-angle-range, in scanner coordinates, to x-y-z locations. These 3D points are the basis for all further processing.

We have two main processing modes: obstacle detection and terrain analysis. Obstacle detection starts by calculating surface normals from the x-y-z points. Flat, traversable surfaces will have vertical surface normals. Obstacles will have surface patches with normals pointed in other directions. This analysis is relatively fast, running in about 5 seconds on a Sun 3/75, and is adequate for smooth terrain with discrete obstacles.

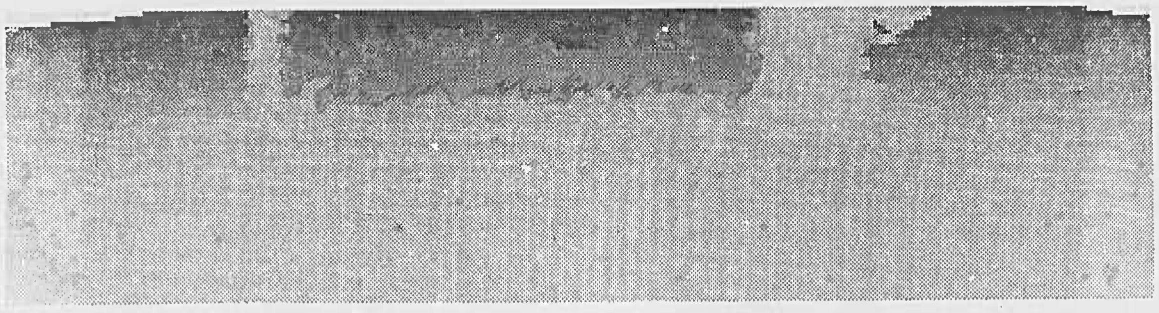


Figure 12. Range image of two trees on flat terrain. Gray levels encode distance; nearer points are painted lighter.

Simple obstacle maps are not sufficient for detailed analysis. For greater accuracy we do more careful terrain analysis and combine sequences of images corresponding to overlapping parts of the environment into an *extended obstacle map*. The terrain analysis algorithm first attempts to find groups of points that belong to the same surface, and then uses these groups as seeds for the region growing phase. Each group is expanded into a smooth connected surface patch. The smoothness of a patch is evaluated by fitting a surface, plane or quadric. In addition, surface discontinuities are used to limit the region growing phase. The complete algorithm is:

1. Edges: Extract surface discontinuities, pixels high with jumps in x-y-z.

2. Clustering: Find clusters in the space of surface normals and identify the corresponding regions in the
3. Region growing: Expand each region until the fitting error is larger than a given threshold. The expansion proceeds by recursively adding the point of the region boundary that adds the minimum fitting error.

The clustering step is designed so that other attributes such as color or curvature can also be used to find potential regions on the object. The primitive surface used to compute the fitting error can be either a plane or a quadric surface. The decision is based on the size of the region.

Obstacle detection works at longer range than terrain analysis. When the scanner is looking at distant objects, it has a very shallow depression angle. Adjacent scanlines, separated by 1/2 degree in the range image, can strike the ground at widely different points. And since the grazing angle is shallow, very little of the emitted laser energy returns to the sensor, producing noisy pixels. Noisy range values, widely spaced, make it difficult to do detailed analysis of flat terrain. But a vertical obstacle, such as a tree, shows up much better in the range data. Pixels from neighboring scanlines fall more closely together, and with a more nearly perpendicular surface the returned signal is stronger and the data is cleaner. So it is much easier for obstacle detection to find obstacles than for terrain analysis to certify that a patch of ground is smooth and level.

There are cases in which neither video nor range alone provide enough information, where we have to do data fusion to determine mobility or recognize an object. One such example occurs in navigating the smaller Terregator vehicle around campus sidewalks. At one spot, a sidewalk goes up a flight of stairs and a bicycle path curves around. Video alone has a tough time distinguishing between the cement stairs and the cement bicycle path. Range data cannot tell the difference between the smooth rise of the grassy hill and the smooth bicycle ramp. The only way to correctly identify the safe vehicle path is to use both kinds of data.

We start by fusing the data at the pixel level. For each range point, we find the corresponding pixel in the video image. We produce a painted range image, in which each pixel is a {red, green, blue, x, y, z} 6-vector. Then we can run our standard range segmentation and color segmentation programs, producing regions of smooth range or constant color. For the stairs in particular, we have a special-purpose step detection program that knows about vertical and horizontal planes, and how they are related in typical stairs. It is easy to combine the regions from these separate processes, since they are all in the same coordinates of the painted range image. The final result is a smooth concrete region, in which it is safe to drive, and a positive identification and 3D location of the stairs, for updating the vehicle position.

4. System Building

Besides the problems in building the various modules for road finding, obstacle detection, path planning, and so forth, there are also a set of issues related to building a coherent system. We have built a layered system, with CODGER providing tools and services, and an architecture on top setting conventions for control and data flow.

4.1 Blackboards and Whiteboards

The program organization of the NAVLAB software is shown in Figure 13. Each of the major boxes represents a separately running program. The central database, called the *Local Map*, is managed by a program known as the Local Map Builder (*LMB*). Each module stores and retrieves information in the database through a set of subroutines called the *LMB Interface* which handle all communication and synchronization with the LMB. If a module resides on a different processor than the LMB, the LMB and LMB Interface will transparently handle the network communication. The Local Map, LMB, and LMB Interface together comprise the CODGER (COmmunications Database with GEometric Reasoning) system.

The overall system structure--a central database, a pool of knowledge-intensive modules, and a database manager that synchronizes the modules--is characteristic of a traditional blackboard system. Such a system is called "heterarchical" because the knowledge is scattered among a set of modules that have access to data at all levels of the database (i.e., low-level perceptual processing ranging up to high-level mission plans) and may post their findings on any level of the database; in general, heterarchical systems impose *de facto* structuring of the information flow among the modules of the system. In a traditional blackboard, there is a single flow of control managed by the database (or blackboard) manager. The modules are subroutines, each with a pre-determined precondition (pattern of data) that must be satisfied before that module can be executed. The manager keeps a list of which modules are ready to execute, and in its central loop it selects one module, executes it, and adds to its ready-list any new modules whose preconditions become satisfied by the currently executing module. The system is thus synchronous and the manager's function is to focus the attention of the system by selecting the "best" module from the ready-list on each cycle.

We call CODGER a *whiteboard* because it also implements a heterarchical system structure, but differs from a blackboard in several key respects. In CODGER, each module is a separate, continuously running program; the modules communicate by storing and retrieving data in the central database. Synchronization is achieved by primitives in the data retrieval facilities that allow, for example, for a module to request data and suspend execution until the specified data appears. When some other module stores the desired data, the first module will be re-activated and the data will be sent to it. With CODGER a module programmer thus has control over the flow of execution within his module and may implement real-time loops, demons, data flows among cooperating modules, etc. CODGER also has no pre-compiled list of data retrieval specifications; each time a module requests data, it provides a pattern for the data desired at that time. We call such a system a whiteboard--it is heterarchical like a blackboard, but each module runs in parallel with the module programmer controlling the synchronization and data retrieval requests as best suited for each module. Like other recent distributed AI architectures, whiteboards are suited to execution on multiple processors.

4.2 Data Storage and Retrieval

Data in the CODGER database (Local Map) is represented in *tokens* consisting of classical *attribute-value pairs*. The types of tokens are described in a *template file* that tells the name and type of each attribute in tokens of each type. The attributes themselves may be the usual scalars (integers, floating-point values, strings, enumerated types), arrays (or sets) of these types (including arrays of arrays), or geometric locations as described below. CODGER automatically maintains certain attributes for each token: the token type and id number, the "generation number" as the token is modified, the time at which the token was created and inserted into the database, and the time at which the sensor data was acquired that led to the creation of this token. The LMB Interface provides facilities for building and dissecting tokens and attributes within a module. Rapid execution is supported by mapping the module programmer's names for tokens and attributes onto globally used index values at system startup time.

A module can store a token by calling a subroutine to send it to the LMB. Tokens can be retrieved by constructing a pattern called a *specification* and calling a routine to request that the LMB send back tokens matching that specification. The specification is simply a boolean expression in which the attributes of each token may be substituted; if a token's attributes satisfy the boolean expression, then the token is sent to the module that made the request. For example, a module may specify:

tokens with **type** *equal to "intersection" and* **traffic-control** *equal to "stop-sign"*

This would retrieve all tokens whose **type** and **traffic-control** attributes satisfy the above conditions. The specification may include computations such as mathematical expressions, finding the minimum value within an array attribute, comparisons among attributes, etc. CODGER thus implements a general database. The module programmer constructs a specification with a set of subroutines in the CODGER system.

One of the key features of CODGER is the ability to manipulate geometric information. One of the attribute types provided by CODGER is the *location*, which is a 2-D- or 3-D polygon and a reference to a *coordinate frame* in which that polygon is described. Every token has a specific attribute that tells the location of that object in the Local Map, if applicable, and a specification can include geometric calculations and expressions. For example, a specification might be:

tokens with **location** *within 5 units of (45,32) [in* world *coordinates]*

or

tokens with **location** *overlapping X*

where *X* is a description of a rectangle on the ground in front of the vehicle. The geometric primitives currently provided by CODGER include calculation of centroid, area, diameter, convex hull, orientation, and minimum bounding rectangle of a location, and distance and intersection calculations between a pair of locations. We believe that this kind of geometric data retrieval capability is essential for supporting spatial reasoning in mobile robots with multiple sensors. We expect geometric specifications to be the most common type of data retrieval request used in the NAVLAB.

CODGER also provides for automatic coordinate system maintenance and transformation for these geometric operations. In the Local Map, all coordinates of location attributes are defined relative to **WORLD** or **VEHICLE** coordinates; **VEHICLE** coordinates are parameterized by time, and the LMB maintains a time-varying transformation between **WORLD** and **VEHICLE** coordinates. Whenever new information (i.e., a new **VEHICLE-to-WORLD** transform) becomes available, it is added to the "history" maintained in the LMB; the LMB will interpolate to provide intermediate transformations as needed. In addition to these basic coordinate systems, the LMB Interface allows a module programmer to define *local coordinates* relative to the basic coordinates or relative to some other local coordinates. Location attributes defined in a local coordinate system are automatically converted to the appropriate basic coordinate system when a token is stored in the database. CODGER provides the module programmer with a conversion routine to convert any location to any specified coordinate system.

All of the above facilities need to work together to support asynchronous sensor fusion. For example, suppose we have a vision module A and a rangefinder module B whose results are to be merged by some module C. The following sequence of actions might occur:

1. A receives an image at time 10 and posts results on the database at time 15. Although the calculations were carried out in the camera coordinate system for time 10, the results are automatically converted to the VEHICLE system at time 10 when the token is stored in the database.

2. Meanwhile, B receives data at time 12 and posts results at time 17 in a similar way.

3. At time 18, C receives A's and B's results. As described above, each such token will be tagged with the time at which the sensor data was gathered. C decides to use the vehicle coordinate system at time 12 (B's time) for merging the data.

4. C requests that A's result, which was stored in VEHICLE time 10 coordinates, be transformed into VEHICLE time 12 coordinates. If necessary, the LMB will automatically interpolate coordinate transformation data to accomplish this. C can now merge A's and B's results since they are in the same coordinate system. At time 23, C stores results in the database, with an indication that they are stored in the coordinate system of time 12.

4.3 Synchronization Primitives

CODGER provides module synchronization through options specified for each data retrieval request. Every time a module sends a specification to the LMB to retrieve tokens, it also specifies options that tell how the LMB should respond with the matching tokens:

- *Immediate Request.* The module requests all tokens currently in the database that match this specification. The module will block (i.e., the "request" subroutine in the LMB Interface will not return control) until the LMB has responded. If there are no tokens that match the specification, the action taken is determined by an option in the module's request:
  - o *Non-Blocking.* The LMB will answer that there are no matching tokens, and the module can then proceed. This would be used for time-critical modules such as vehicle control. Example: "Is there a stop sign?"
  - o *Blocking.* The LMB will record this specification and compare it against all incoming tokens. When a new token matches the specification, it will be sent to the module and the request will be satisfied. Meanwhile, the module will remain blocked until the LMB has responded with a token. This is the type of request used for setting up synchronized sets of communicating modules: each one waits for the results from the previous module to be posted to the database. Example: "Wake me up when you see a stop sign."
- *Standing Request.* The module gives a specification along with the name of a subroutine. The module then continues running; the LMB will record the specification and compare it with all incoming tokens. Whenever a token matches, it will be sent to the module. The LMB Interface will intercept the token and execute the specified subroutine, passing the token as an argument. This has the effect of invoking the given subroutine whenever a token appears in the database that matches the given specification. It can be used at system startup time for a module programmer to set up "demon" routines within the module. Example: "Execute that routine whenever you see a stop sign."

4.4 Architecture

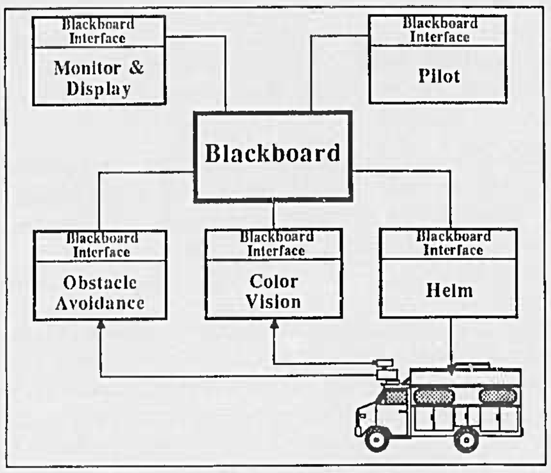


Figure 13. Navlab software architecture.

There are several modules that use the CODGER tools, and that fit into a higher level architecture. The modules are:

- Pilot. Looks at the map and at current vehicle position to predict road location for Vision. Plans paths.
- Color Vision. Waits for a prediction from the Pilot, waits until the vehicle is in the best position to take an image of that section of the road, returns road location.
- Obstacle Avoidance. Gets a request from the Pilot to check a part of the road for obstacles. Returns a list of obstacles on or near that chunk of the road.
- Helm. Gets planned path from Pilot, converts polyline path into smooth arcs, steers vehicle.
- Graphics and Monitor. Draws or prints position of vehicle, obstacles, predicted and perceived road.

There are three other modules in our architecture but not yet implemented:

- Captain. Talks to the user and provides constraints such as "avoid area A" or "go by road B".
- Map Navigator. Maintains a map, does global path planning, provides long-term direction to the Pilot.
- Lookout. Looks for landmarks and objects of importance to the mission.

These modules use CODGER to pass information about "driving units". A driving unit is a short chunk of the road or terrain (in our case 4 meters long) treated as a unit for perception and path planning. The Pilot gives driving unit predictions to Color Vision, which returns an updated driving unit location. Obstacle Detection then sweeps a driving unit for obstacles. the Pilot takes the driving unit and obstacles, plans a path, and hands the path off to the Helm. The whole process is set up as a pipeline, in which Color Vision is looking ahead 3 driving units, Obstacle Detection is looking 2 driving units ahead, and path planning at the next unit. If for any reason some stage slows down, all following stages of the pipeline have to wait. So, for instance, if Color Vision is waiting for the vehicle to come around a bend so it can see down the road, Obstacle Detection will finish its current unit and will then have to wait for Color Vision to finish. In an extreme case, the vehicle may have to come to a halt until everything clears up. All planned paths include a deceleration to a stop at the end, so if no new path comes along to overwrite the current path the vehicle will stop before driving into an area that has not been seen or cleared of obstacles.

In our current system and test area, three driving units is too far ahead for Color Vision to look, so both Color Vision and Obstacle Detection are looking at the same driving unit. Obstacle Detection looks at an area enough larger than the Pilot's predicted driving unit location that the actual road is guaranteed to be covered. Another practical modification is to have Obstacle Detection look at the closest driving unit also, so if a person walks onto the road immediately in front of the vehicle he will be noticed. Our system will try to plan a path around obstacles while remaining on the road. If that is not possible, it will come to a halt and wait for the obstacle to move before continuing.

5. Conclusions and Future Work

The system described here works. Since quashing the last of our (known) bugs, it has successfully driven the Navlab many tens of times, processing thousands of color and range images without running off the road or hitting any obstacles. CODGER has proved to be a useful tool, handling a lot of the details of communications and geometry. Module developers have been able to build and test their routines in isolation, with relatively little integration overhead. Yet there are several areas that need much more work.

Speed. We drive the Navlab at 10 cm/sec, a slow shuffle. Part of the reason for this is because our test road is narrow and winding, and part of the reason is that we deliberately concentrate on competence rather than speed. We are pursuing several ways of increasing speed. One bottleneck is the computing hardware. We are mounting a Warp, CMU's experimental high speed processor, on the Navlab. The Warp will give us a factor of 100 more processing power than a Sun for color and range image processing. At the same time, we are looking at improvements to the software architecture. We need a more sophisticated path planner, and we need to process images that are more closely spaced than the length of a driving unit. Also, as the vehicle moves more quickly, our simplifying assumption that steering is instantaneous and that the vehicle moves along circular arcs becomes more seriously flawed. We are looking at other kinds of smooth arcs, such as clothoids.

Map. One particular reason for the slow speed is that the Pilot assumes straight roads. We need to have a description that allows for curved roads, with some constraints on maximum curvature. The next steps will include building maps as we go, so that subsequent runs over the same course can be faster and easier.

Cross country travel. Travel on roads is only half the challenge. The Navlab should be able to leave roads and venture cross country. Our plans call for a fully integrated on-road/off-road capability.

Intersections. Current vision routines have a built in assumption that there is one road in the scene. When the Navlab comes to a fork in the road, vision will report one or the other of the forks as the true road depending on which looks bigger. It will be important to extend the vision geometry to handle intersections as well as straight roads. We already have this ability on our sidewalk system, and will bring that over to the Navlab.

Landmarks. Especially as we venture off roads, it will become increasingly important to be able to update our position based on sighting landmarks. This involves map and perception enhancements, plus understanding how to share limited resources, such as the camera, between path finding and landmark searches.

Software Development. Our current blackboard system can manipulate primitive data elements but has no concept of data structures made up of tokens on the blackboard. We need this for representing complex 3D geometric descriptions of objects for recognition. We will also be implementing a LISP interface to our blackboard so that not all modules need to be written in C.

Integration with Work from Other Sites. .There are other universities and corporations cooperating with Carnegie-Mellon through DARPA's Strategic Computing Vision program. We plan to incorporate some of their programs into the Navlab system in the coming years as it evolves into the "New Generation Vision System" which is that goal of that program.

Acknowledgements

This research is supported in part by the Advanced Research Projects Administration of the Department of Defense (DARPA), under contract DACA 76 85-C-003, monitored by the US Army Engineer Topographic Laboratories. We are part of the Strategic Computing Vision effort.

The Terregator and Navlab were built by William Whittaker's group in the Construction Robotics Laboratory, and the Warp group is led by H. T. Kung. Faculty in the ALV group include Martial Hebert (range finder work), Jon Webb (Warp vision), and Steve Shafer (CODGER). The real work gets done by an army of 8 staff, 9 graduate students, 5 visitors, and 3 part time programmers.

Copyright 1987 The Robotics Institute, Carnegie Mellon University.

References

[1] J. Crisman.
Machine Perception.
Unix Review 4(9), 1986.

[2] Elfes, A.
A Sonar-Based Mapping and Navigation System.
In *IEEE International Conference on Robotics and Automation*. 1986.

[3] Y.Goto, K,Matsuzaki, I.Kweon, T.Obatake.
CMU Sidewalk Navigation System.
In *Fall Joint Computer Conference*. ACM/IEEE, November, 1986.

[4] Hebert, M., and Kanade, T.
Outdoor Scene Analysis Using Range Data.
In *IEEE International Conference on Robotics and Automation*. 1986.

[5] Kanade, T., Thorpe, C., and Whittaker, W.
Autonomous Land Vehicle Project at CMU.
In *ACM Computer Conference*. Feb, 1986.

[6] Takeo Kanade and Charles Thorpe.
CMU Strategic Computing Vision Project Report: 1984 to 1985.
Technical Report, The Robotics Institute, Carnegie-Mellon University, 1985.

[7] Krogh, B., and Thorpe, C.
Integrated Path Planning and Dynamic Steering Control for Autonomous Vehicles.
In *IEEE International Conference on Robotics and Automation*. 1986.

[8] L.H. Matthies and S.A. Shafer.
Error modelling in stereo navigation.
In *Fall Joint Computer Conference*. ACM/IEEE, November, 1986.

[9] B. Serey and L. Matthies.
Obstacle avoidance using 1-D stereo vision.
Technical Report, Carnegie Mellon Robotics Institue, 1987.

[10] Shafer, S., Stentz, A., Thorpe, C.
An Architecture for Sensor Fusion in a Mobile Robot.
In *IEEE International Conference on Robotics and Automation*. 1986.

[11] Jeff Singh et al.
NavLab: An Autonomous Vehicle.
Technical Report, Carnegie Mellon Robotics Institute, 1986.

[12] C. Thorpe.
Vision and Navigation for the CMU Navlab.
In *SPIE*. Society of Photo-Optical Instrumentation Engineers, October, 1986.

[13] Wallace, R., Matsuzaki, K., Goto, Y., Crisman, J., Webb, J., and Kanade, T.
Progress in Robot Road-Following.
In *IEEE International Conference on Robotics and Automation*. 1986.

VISION-BASED NAVIGATION: A STATUS REPORT

Larry S. Davis
Daniel Dementhon
Ramesh Gajulapalli
Todd R. Kushner
Jacqueline Le Moigne
Phillip Veatch

Center for Automation Research
University of Maryland
College Park, MD 20742

ABSTRACT

This report describes research performed during the first two years on the project *Vision-Based Navigation for Autonomous Vehicles* being conducted under DARPA support. Our project, to date, has focused on four goals: development of a vision system for autonomous navigation of roads and road networks; support of Martin Marietta Aerospace, Denver, the integrating contractor on DARPA's ALV program; experimenting with the vision system developed at Maryland on the Martin Marietta ALV; and development and implementation of parallel algorithms for visual navigation. In this paper we will restrict our attention to the first of these four goals.

1. INTRODUCTION

This report describes research performed during the first two years on the project *Vision-Based Navigation for Autonomous Vehicles* being conducted under DARPA support. Our project, to date, has focused on four goals:

1) Development of a vision system for autonomous navigation of roads and road networks.
2) Support of Martin Marietta Aerospace, Denver, the integrating contractor on DARPA's ALV program. This is done principally through a set of small Vision Working Groups that include representatives from Maryland, Martin Marietta, Carnegie-Mellon University and relevant subsets of the Vision Technology Base contractors of the Strategic Computing Program.
3) Experimenting with the vision system developed at Maryland on the Martin Marietta ALV.
4) Development and implementation of parallel algorithms for visual navigation on the parallel computers developed under the DARPA Strategic Computing program—specifically, the WARP systolic array processor, the Butterfly and the Connection Machine.

In this paper we will restrict our attention to the first of these four goals. An extended version of this overview, as well as descriptions of the work performed in support of the other goals of the project, can be found in Davis et al.[1]. This paper also does not include references to the extensive literature on autonomous navigation. Those references, as well as discussions of the relationships between our work and the work of others, are included in the reports cited in this paper.

We have constructed, in our laboratory, a simulation facility for developing our visual navigation system. This facility consists of a robot arm carrying a sensor package over a terrain board. Roads and road networks are painted on the terrain board. The sensor package includes a black and white solid state camera, a structured light range scanner that provides range data registered with the black and white video data, and position encoders that allow us to control the height and attitude of the image sensors with respect to the terrain board. The simulation facility is described in greater detail in Dementhon et al.[2]. While the imagery produced by the sensors on the robot arm is quantitatively different from data collected by the sensors on the ALV at Martin Marietta, we have found this simulator to be very valuable in designing the control components of the navigation system; these components account for the larger part of the actual code in the system.

We have also developed a software simulation system for range data analysis. This is based on a motion simulation program developed to support experimentation in dynamic stereo (Waxman[3]). The simulator generates range images of a three dimensional polyhedral world. It is described in Veatch and Davis[4].

Section 2 of this report provides an overview of our visual navigation system. Figure 1 is a block diagram of the principal components of that system. The organization of the system was originally described in Waxman et al.[5].

The vision executive controls the activities of the other modules in the system. Its principal responsibilities are

1) Controlling the focus of attention mechanism for image processing (i.e., identifying subsets of the sensed data likely to contain critical features for navigation and predicting both the photometric and geometric properties of these features).
2) Using knowledge stored in its visual knowledge base both to verify the model of its environment con-

structed on the basis of analyzing previous images and to extend that model further in the direction of travel.

The vision executive is described in more detail in Section 2.2. The image processing module (Section 2.3) contains algorithms for extracting structures from the image data that potentially correspond to objects of interest in the scene, such as boundaries of roads or obstacles on roads. The geometry module (Section 2.4) contains a variety of methods for reconstructing the three-dimensional geometry of the scene. The vision executive chooses the appropriate reconstruction technique either based on *a priori* knowledge about the environment or from its analysis of previously sensed data.

During the past several months we have extended the navigation system so that it can navigate over networks of roads. We have done this by incorporating a map database (Section 2.5) that includes a map of the roads on the terrain board. The organization of this database is identical to that of the road network map compiled by ETL for the Martin Marietta test site. The map is used by an extended scene prediction module (Section 2.6) to provide the vision executive with predictions about the appearance of the road(s) in its field of view. These predictions not only allow the vision executive to control its focus of attention mechanism, but also to control the viewing directions of its sensors to ensure that critical scene features are in their field of view.

Our experience with the system described in Section 2 has made it clear that a more flexible framework is needed, principally for specifying the control aspects of navigation. For example, our current system has a single strategy for tracking the boundaries of the road in an image—the scene prediction module identifies windows near the bottom of the image that will contain the projections of the left and right road boundaries, and after the image processing module identifies the actual locations of the projected road boundaries in those windows subsequent windows are placed that allow us to *continuously* track the road boundaries through the image. However, it would sometimes be advantageous to use some other strategy for tracking the road. For example, if prior knowledge indicated that the stretch of road currently being traversed were straight then one could imagine a strategy where after the initial windows were processed subsequent windows were placed relatively high up in the image, corresponding to segments of road relatively far from the vehicle. These and similar considerations have led us to design and develop an object-oriented system for visual navigation. A description of this system is included in Section 3.

2. VISUAL NAVIGATION SYSTEM

2.1 Introduction

This section contains an overview of the visual navigation system. This system runs on a VAX/VICOM environment, with image acquisition and some low level processing being done on the VICOM and the remainder of the vision processing, planning and navigation being performed on the VAX.

2.2 Vision Executive

The principal responsibilities of the vision executive are:

1) controlling the focus of attention mechanism for sensor data processing, and
2) verifying and extending the sensor-based model of the environment.

Both of these tasks involve sensor control (for example, determining the appropriate pan and tilt for the video cameras), map database analysis and geometric reasoning about the three dimensional properties of scene objects identified in the sensor data.

Consider, for example, the verification of that portion of the scene model not yet traveled—i.e., still in front of the vehicle. Due to limitations of the vision and navigation systems this model will, to some degree, be inaccurate. Both to reduce these inaccuracies and to provide an anchor for analyzing previously unexplored parts of the scene, the vision executive analyzes those parts of the sensed data expected to contain components of the scene model. In order to do this, the vision executive must first decide, before acquiring an image, what parts of the scene model it will search for. Then, based on the three dimensional properties of the scene model and a model for the assumed motion of the vehicle through that model, it can determine a viewing angle for the camera that will produce an image that will include those components. Currently this is done by trying to center the field of view of the camera on a point in the center of the road a fixed distance in front of the vehicle. The distance chosen is a function of both vehicle speed and the extent of the current three dimensional road model. One could imagine, of course, more sophisticated strategies for camera control based on other visual goals.

Once the direction of view of the camera is determined, the vision executive can work with the scene predictor to establish the projected positions of key scene features in the next image frame. In the current implementation, the following types of predictions are supported:

1) Identify the points on the boundary of the image where the left and right road edges will enter the image.
2) Given a point x feet in front of the vehicle on the left (right) boundary of the road, will that point appear in the image and, if so, where?

The vision executive then places rectangular windows around these predictions and sends those windows to the appropriate sensor data processing module to extract the corresponding image object (straight edge segments of specific orientation and contrast sign for identifying road boundaries).

Currently, the "verification" stage of processing ends after the first two windows (one on the left boundary and one on the right boundary of the road) are processed since it is only for these two windows that expectations are generated concerning image properties such as orientation and contrast. Both to place and to analyze subsequent windows the vision executive applies either information derived from a road map, or, in the absence of a map, general knowledge about the geometry of roads, to generate subsequent predictions about the image locations of road features.

In the case where map data is available, the vision executive has available to it a set of precomputed predictions, indexed by world road coordinates, concerning the structure of the road in the immediate vicinity of the vehicle. Due to uncertainties in vehicle position and the limited accuracy of such map information the vision executive can only use this information in a qualitative fashion. For example, if the map database indicates that the road will be turning to the right, then the linear segments extracted by the image processing should turn towards the right in the image. Of more interest is the case where the map indicates that the vehicle is approaching an intersection. The images of intersections are quite complex, and the vision executive attempts, initially, to avoid processing the specific part of the image where the intersecting roads actually meet until it has identified parts of the image containing the constituent roads. It does this by identifying windows predicted to include the intersection, and then searching around these windows for the intersecting roads. The information potentially available to the vision executive concerning the structure of the intersection includes the spectral properties of the roads, their widths, and the angles between the roads at the intersections.

2.3 Image Processing

2.3.1 Video image processing

The Image Processing module transforms an input image into a symbolic representation of the boundaries of the roads in the field of view by extracting dominant linear features from grey-level or color imagery. Different representations can be used in the image domain: boundary-based and region-based are two examples of such representations. We present two kinds of algorithms for extracting roads from imagery, corresponding to these two different representations: linear feature extraction and grey-level or color segmentation.

2.3.1.1 Thresholding algorithms

In this section we present algorithms relying primarily on segmenting grey-scale and color images to locate dominant linear features in the feed-forward mode of operation. The first uses a prediction of where the road boundaries will appear in small windows at the bottom of the image, along with a boundary-based representation of the linear features in those windows, to collect statistics of grey scale or color values on the surface of the road.

Initial windows covering segments of the left and right boundaries of the road are chosen based on projecting the current 3-D road model onto the image plane and determining where the road boundaries enter the image. For these windows we estimate the orientation and position of the projection of the road edges using a Hough transform as described in Davis et al. [6]. After the projection of the road edges are determined in each window, the two regions of each window separated by those edges, presumably road and non-road, are histogrammed separately. A minimum-error threshold for each window is then computed from these histograms. The thresholds are then each applied to half of the image to segment the image into *road pixels* and *non-road pixels*.

Processing differs for subsequent windows by constraining the lines in these windows to connect to the lines in the immediately preceding windows (the *road continuity* assumption)—e.g., an endpoint of the line in the previous window becomes the *pivot*, or intercept, of the line in the current window. Furthermore, we constrain the orientation of the line in the current window to be in a small interval, $[\theta_m, \theta_M]$, centered about the orientation of the line in the previous window. Since the pivot point is fixed, we need only estimate one parameter—the direction, θ, of the line through the pivot point. Given the pivot point, (x_p, y_p) and any road point (x_v, y_v) in the window, the angle θ of the road point with respect to the pivot is simply

$$\theta = \tan^{-1}((y_v - y_p) / (x_v - x_p)).$$

If the values of θ for *road pixels* in a window are accumulated, we would expect the counts to suddenly drop off at a value of θ roughly corresponding to the line segment best fitting the road boundary. The angle corresponding to the accumulator value closest to a fixed threshold is chosen as the boundary line. To prevent the line from overshooting beyond the actual road boundary, the line is cut back to the road point furthest along the line from the pivot. Figure 2 shows the algorithm applied to an image in feed-forward mode.

Since the determination of the orientation and position of the projection of the road edges in the initial and subsequent windows using the Hough transform is expensive, a second algorithm was developed that relies only on the initial windows containing portions of road and non-road around the projection of the current 3-D road model onto the image. A threshold is calculated for each window by sampling two populations of the window assumed to contain only road and non-road pixels, in two small diagonally opposite corners of the window, and choosing a threshold satisfying a minimum-error criterion for the two samples. The left and right boundaries of the road in the image are extracted using the thresholds. Line segments are fit to the border by determining for each window the road/non-road border point along the top, middle, and bottom window rows that satisfies the following minimum-error criterion: the total of the fraction of road points on its non-road side of the window row and the fraction of non-road points on its road side of the window row is at a minimum. Figure 3 shows some results of this algorithm in feed-forward mode.

Contrast reversal can occur across the road boundary, causing simple thresholding segmentation to fail. A third algorithm calculates thresholds by sampling a population of the window assumed to only contain road pixels, in the lowest corner closest to the center of the road, and selecting a range of threshold values centered about the mean of the sample of size equal to a constant number of standard deviations of the sample. Figure 4 shows the algorithm applied to an image that exhibits contrast reversal.

2.3.1.2 Multiresolution algorithms

To improve region-based segmentation of grey scale anu color images, the image can be blurred to reduce the effects of background texture and minor variations on the surface of the road. A better approach, however, is to reduce the image size if possible. In our experiments, good segmentations were obtained for road images of size 64 by 64 pixels. Another improvement over the above methods that select left and right boundary road edges separately is to select the left and right road edges simultaneously constrained by knowledge about the geometric properties of the road, e.g., road width.

This algorithm for locating dominant linear features in an image at different resolutions in the feed-forward mode of operation uses knowledge about the predicted location of the road to position a window at the lower middle part of the road assumed to contain only road pixels. Statistics of grey scale or color in this window are computed and a range of threshold values are selected about the mean of the sample of size equal to a constant number of standard deviations of the sample. The entire image is segmented using these threshold values. An initial window covering segments of both the left and right boundaries of the road is chosen based on projecting the current 3-D road model onto the image plane and determining where the road boundaries enter the image. Line segments are fit to the borders by simultaneously determining for the window the road/non-road border points along the top, middle, and bottom window rows that satisfy two minimum-error criteria: the total of the fraction of road points outside the border points and the fraction of non-road points between the border points is at a minimum, and the distance between the border points is closest to the predicted distance between the left and right boundaries of the road projected from the current 3-D road model onto the image at that line.

Once a window is processed, the next window is chosen in such a way that the length of both line segments found in this extrapolated window is maximized. This is currently done by placing the next window so that its center lies on the road centerline projected from the previous window. Processing is automatically stopped when the window is within 20% of the top of the image, when the window leaves the image, or when no left or right window border points are found. Figure 5 shows the results of applying this algorithm to a road image from the Martin Marietta test site.

2.3.2 Range data processing

A surface may be considered navigable if its slope is sufficiently small. In the extreme case of an obstacle sticking up from the ground the area will contain a discontinuous slope that, when digitized, will appear as a very steep slope and hence mark the obstacle as being an unnavigable area. The slope is measured by calculating the first derivatives of the range with respect to the vertical and horizontal scanning angles. These calculations involve nothing more than addition and subtraction and so can be performed very quickly. Obstacle detection from first derivatives has the desirable property of being less sensitive to uncertainties in the range scanner's orientation than algorithms based on the range itself.

The range scanner is mounted on the ALV approximately nine feet above the ground and looks out in the direction that the vehicle is travelling. An ERIM range image is most naturally described in a spherical coordinate system in which the 64 rows of the image are at equally spaced values of the vertical scan angle, ϕ, and the 256 columns are at evenly spaced values of the horizontal scan angle, θ. The image has a 30 degree vertical field of view ranging from approximately 6 degrees beneath the horizon to 36 degrees beneath the horizon. The 80 degree horizontal field of view extends from about 130 degrees to 50 degrees as measured from the x-axis.

The range at each pixel in a range image is calculated in hardware from the wave phase difference of a modulated laser beam. This causes the calculated ranges to be the true range modulo 64 feet; i.e., a value of 10 feet may indicate that the signal is returning from 10 feet, 74 feet, 138 feet, etc. The resultant ambiguity must be compensated for by any obstacle detection algorithm.

Due to the ambiguity effect, output range values are all between 0 and 64 feet. They are quantized into three inch units so that the final output of the scanner is a 64×256 array of 8 bit values ranging from 0 to 255.

The geometry of the range scanner results in the following range-height relationship:

$$Y = \rho \sin(\theta) \sin(\phi) \tag{1}$$

Hence

$$\frac{\partial \rho}{\partial \theta} = \frac{-Y}{\sin(\theta)\tan(\theta)\sin(\phi)} \tag{2}$$

$$\frac{\partial \rho}{\partial \phi} = \frac{-Y}{\sin(\theta)\tan(\phi)\sin(\phi)} \tag{3}$$

We estimate the θ derivative at the (i,j)th pixel in the image by averaging over a 3×3 neighborhood

$$\frac{\partial \rho}{\partial \theta} \approx \sum_{k=i-1}^{i+1} Range[k][j+1] - \sum_{k=i-1}^{i+1} Range[k][j-1] \tag{4}$$

Similarly, the ϕ derivative is calculated as

$$\frac{\partial \rho}{\partial \phi} \approx \sum_{k=j-1}^{j+1} Range[i+1][k] - \sum_{k=j-1}^{j+1} Range[i-1][k] \qquad (5)$$

One could simply threshold these images and assume that large changes in the absolute values of the derivatives indicate surfaces that are rapidly changing orientation and hence are not navigable. The problem with this approach is that it severely reduces one's ability to detect small obstacles. A perfectly flat surface will yield an estimated ϕ derivative of about -30 if it is 60 feet away but the same surface at 10 feet only has a ϕ derivative of about -.9 (these values are the unnormalized sums of the 3×3 neighborhood calculation shown in equation (5)). The solution is to threshold the image at varying levels depending on the distance of the object. In practice we do this by creating an image of synthetic derivatives based on a flat surface and subtracting this from the actual derivative image.

Large absolute values of the ϕ derivative indicate obstacle edges and ground surfaces that are sloping rapidly down away from the ALV. As a surface becomes more vertical its ϕ derivatives will become less negative and actually cross over into positive values as the surface approaches an upright orientation. These values (which are relatively small in comparison to edge jumps or steep downward slopes) would be lost if thresholded with the rest of the image. For this reason we segment the ϕ derivative image into positive and negative regions and threshold the positive values independently.

Once an obstacle is detected one wishes of course to know where it is. This raises another interesting problem arising from the manner in which the range was found by the scanner. As the laser beam goes out from the scanner it diverges in the shape of a cone. The return signal that determines how far the beam traveled is a sum of all of the ranges within the cone. For example, if half of the cone hits a tree and the other half hits the ground behind the tree the returning signal would yield a value that is somewhere between the distance to the tree and the longer distance to the ground. This is called the mixed pixel problem.

To counter this effect we try to avoid mixed pixels that occur at the edges of obstacles in estimating object location and use the more accurate range values in an object's interior area. The location of the interior is determined by the sign of the derivative. If an obstacle is sticking up from the ground then a negative θ derivative indicates that the pixel is on the left edge of an obstacle so the range is taken from the pixel that is to the right of the current pixel. The same approach can be used with the ϕ derivative for determining if one is at a top (negative derivative) or bottom (positive derivative) edge. We have found that this strategy works well for avoiding false ranges due to mixed pixels.

The ambiguity interval problem has been approached in two different ways. For actual range images taken from the ALV we have found that if the vehicle is not travelling over very hilly territory, simply deleting the upper few rows of the image removes most of the pixels that are beyond the first ambiguity interval ($\geq$64 feet). This does not affect the obstacle detection algorithm significantly because these pixels tend to be very mixed anyway (the beam has spread out into a relatively wide cone by the time it has travelled beyond 50-60 feet) and so are of little use for accurate obstacle detection.

A more time consuming but somewhat more precise approach has been to examine each column in the image individually. Whenever adjacent pixels go suddenly from large values to very small values it is reasonable to assume that an ambiguity interval has been reached and that all pixels in the column beyond this point should have 256 added to their ranges. This approach was used in our laboratory-scale range image synthesizer.

Uncertainties in the ALV's orientation with respect to the ground introduces errors into calculations of ranges and range derivatives. We have analyzed these errors and shown that the derivative approach is significantly less sensitive to these uncertainties than the flat earth range method which would simply compute the (absolute) difference between observed and predicted ranges. A summary of this analysis is shown in Table 1.

Figure 6 shows a montage of four range images. The images were produced by an ERIM range scanner mounted on the ALV at Martin-Marietta's Denver test track site. Moving down from the top of the figure, each image is taken five feet further down the road. The person in the last image moved to his position after the first three images were taken and so does not appear in the first three images.

Figures 6a and 6b are, respectively, the thresholded ϕ and θ derivatives. Figure 6c shows the thresholded positive ϕ derivatives. Notice how the interior of the obstacles are well marked in the positive derivative image.

2.4 Geometry Module

Consider the image of the road, which has been processed so that the curves which are images of the two road edges have been found (Figure 7). The Geometry module reconstructs from these two image curves two world curves corresponding to the edges of the world road.

Consider a point p_i on one edge image. The world point of the road edge is a point P_i somewhere along the line Op_i, and the vector P_i is $m_i p_i$; the parameter m_i is the unknown which, once found, will define the world point P_i.

Given a world point P_i on one world edge, there is a corresponding point P'_i on the other world edge such that $P_i P'_i$ determines a line normal to the road centerline. P'_i will be called the *opposite* point of P_i, and the segment $P_i P'_i$ a *cross-segment* of the road. The image p'_i of P'_i is somewhere on a road image edge (the one to which p_i does not belong). The position of p'_i can be defined on this edge in terms of a yet unknown curvilinear abscissa s_i,

$$p'_i = p'_i(s_i),$$

and the world point P'_i which belongs to the line Op'_i can be defined by writing the vector P_i in terms of two parameters,

$$P'_i = m'_i p'_i(s_i),$$

where s_i defines the position of p'_i on the image edge, and m'_i defines the position of P'_i on the line Op'_i.

We can now consider two pairs of opposite world points (P_{i-1}, P'_{i-1}) and (P_i, P'_i). Requiring the segments $P_{i-1}P'_{i-1}$ and $P_iP'_i$ to be cross-segments of the road is equivalent to saying that they are *normal to the centerline*, or that their average direction is normal to the line joining their mid-points. This is written

$$(P'_{i-1} - P_{i-1} + P'_i - P_i)(P_{i-1} + P'_{i-1} - P_i - P'_i) = 0$$

or

$$P_{i-1}P'^2_i = P'_{i-1}P_i^2$$

In other words, *the two diagonals* of the convex quadrilateral formed by $P_{i-1}P'_{i-1}$ and $P_iP'_i$ *must be equal.*

Instead of imposing strict equality, we can try to minimize the difference between the terms while keeping in mind that the points also have to satisfy other constraints:

$$E_{1i} = (P_{i-1}P'^2_i - P'_{i-1}P_i^2)^2$$

We also look for a road with an approximately constant width, and thus look for the points which minimize the quantity

$$E_{2i} = (P_{i-1}P'^2_{i-1} - P'_iP_i^2)^2$$

We could impose the further condition that the cross-segments be more or less horizontal, and thus look for segments which minimize their scalar product with a vertical vector:

$$E_{3i} = (P'_iP_i \cdot \mathrm{V})^2$$

If P_{i-1} and P'_{i-1} are known from a previous calculation step, by setting E_{1i}, E_{2i} and E_{3i} to zero three equations to calculate the three unknowns m_i, m'_i and s_i are obtained. A method of solution ("the zero-bank algorithm") has been proposed to this problem (see Dementhon [7]), for the case when the edge images are chains of line segments; m_i is found as a solution of a cubic equation, and an additional condition of limited slope variation of the road is required to choose among the solutions of this equation. Once the cross-segment P'_iP_i is found, the cross-segment $P'_{i+1}P_{i+1}$ can be determined. The method is an iterative reconstruction of the road, and therefore the first world segment must be known to start the process. Also, we can expect this method to deteriorate from error accumulation, and to perform poorly in case of bank or width variations of the world road. Figure 8 shows an example of applying this algorithm to a synthetic road image.

2.5 Map Database

A map database of the ALV test site has been constructed by the Engineering Topographic Laboratory. This map contains information about roads, topography, surface drainage and landcover relative to the test site. We have constructed a similar map of our terrain board which contains only information about the network of roads. The map contains information about road width, road composition (asphalt, concrete, etc.), road markings (e.g., lane markers), boundary information (presence of road shoulders, properties of the surrounding terrain—e.g., vegetation, sand, etc.) and network topology and geometry. The roads are segmented into pieces which have (nearly) constant properties. So, for example, if a road changes width at some point then that point would be the boundary between two road pieces in the representation of the road. Of course, it is not necessarily the case that all of this information would be provided, *a priori*, in the map database. Some of it may be acquired as the vehicle travels along a road, and some may simply not be available to the vision executive in planning its strategy for identifying the road.

The roadmap is preprocessed by computing, for a regular sampling of points along the roads to be traversed in a mission, properties of the expected appearance of the road network when viewed from approximately those positions. These predictions are used by the vision executive both as a source of guidance for planning the image processing operations to be applied to images and as a source of constraints on the interpretation of those analyses. For example, if it is known that the road has a bright centerline and is on relatively flat terrain, then the vision executive might plan to detect the road by searching for that centerline and to reconstruct the three dimensional structure of the road based on *a priori* estimates of road width and the flat earth model.

2.6 Predictor

Section 2.2 discussed the image-processing strategy applied by the feed-forward algorithm: the image analysis is limited to a succession of windows, the position of a new window being deduced from the road edge element detected in the previous window.

The question which the predictor answers is: where do we place the first windows in the image? The basic idea is to position the first windows by predicting approximately where the road edges are going to be in the image, using the images of the edges detected in a previous image. This procedure is illustrated in Figure 9. There the image plane of the vehicle is shown as a line normal to the camera axis and in front of the lens centers, and the camera moves from position C_1 to position C_2. The change of position of the camera from C_1 to C_2 is available from the navigation system of the vehicle. Consider the image m_1 of a world point M for the camera position C_1. Where is the image m_2 of this point M for the camera position C_2?

The drawing shows the method: from m_1 we must deduce the world point M by an *inverse perspeetive* method. Figure 9 shows a simple Flat Earth assumption by which the ground is assumed planar. M is then the eentral projeetion of m_1 on the ground plane. Note that the whole figure is determined entirely by the following elements:

1. the ground, defined by its distance h from the camera positions and by a normal **n**;
2. the image m_1, defined by a vector $\boldsymbol{C_1 m_1}$
3. the eamera foeal length f.
4. the new camera position C_2, defined by the translation veetor $\boldsymbol{C_1 C_2}$ and the eoordinates of the unit veetors of its reference $(\boldsymbol{i_2}, \boldsymbol{J_2}, \boldsymbol{k_2})$ in the coordinate system of the first camera position C_1.

The unknown is the veetor $\boldsymbol{C_2 m_2}$, easily expressed in terms of the known elements. The fact that M is the projeetion of m_1 in the ground plane determines the veetor $C_1 M$:

$$C_1 M = a_1 C_1 \boldsymbol{m_1},$$

$$C_1 M . \boldsymbol{n} = h,$$

$$a_1 = h / C_1 \boldsymbol{m_1} . \boldsymbol{n},$$

$$C_1 M = h C_1 \boldsymbol{m_1} / C_1 \boldsymbol{m_1} . \boldsymbol{n}$$

$C_2 M$ is now known:

$$C_2 M = C_1 M - C_1 C_2,$$

and m_2 is the projection of M on the image plane at position 2:

$$C_2 \boldsymbol{m_2} = a_2 C_2 M,$$

$$C_2 \boldsymbol{m_2} . \boldsymbol{k_2} = f,$$

Thus

$$a_2 = f / C_2 M . \boldsymbol{k_2}$$

$$\cdot \; C_2 \boldsymbol{m_2} = f C_2 M / C_2 M \cdot \boldsymbol{k_2}$$

Of course, all the vectors have to be expressed in the same reference frame, for instanee the reference frame of C_1. The coordinates of m_2 in the second camera position C_2 are then the scalar products of $\boldsymbol{C_2 m_2}$ in the referenee frame of C_1 with the unit vectors of the reference of C_2 expressed in the reference of C_1, i.e., $C_2 \boldsymbol{m_2} . \boldsymbol{i_2}$ and $C_2 \boldsymbol{m_2} . \boldsymbol{j_2}$.

It would therefore be possible to take all the significant points of image 1 and transform them to points of image 2. However, since the vehiele moved between C_1 and C_2, most of the world points corresponding to the C_1 image points will not fall within the field of view of C_2. Furthermore, all we require are two edge points in the image of C_2 on which to anchor the bottom midpoints of the first proeessing windows. Note that for the windows to fit in the image rectangle, these anchor points must belong to a *reduced rectangle*, obtained by reducing the image reetangle laterally by half the window width, and at the top by the window height, as shown on Figure 10. The simplest approach to find the anchor points would then be to find the intersections j and j' of the predicted road edges with the reduced rectangle (Figure 10b).

This method consists of transforming segments $a_1 b_1, b_1 c_1, \cdots$ of C_1 (Figure 10a) into $a_2 b_2, b_2 c_2$ of C_2 (Figure 10b) until we find an intersection with one of the lower three sides of the reduced rectangles. If an intersection is found for only one edge, the other edge being totally out of the reetangle, a prediction ean be eomputed for only one edge, and the vision exeeutive is informed of this situation.

We will refer to this previous method as *method A*. It is elearly not optimal, because it plaees windows at the lowest possible point in the image; the vehicle might not need to know these lowest parts of the edges, beeause when the processing is over, the vehicle will probably have run *beyond* the corresponding world edges and have no more use for the information. In *method B* deseribed next, we try to position the two anehor points at a higher position; however, if method B fails, as happens when the proposed anchor points fall outside of the reduced rectangle, the system uses method A.

Method B is illustrated in Figure 10c. The drawing represents a top view of the world with two vehicle positions, giving camera positions C_1 and C_2. When the camera reaches C_2, its position with respect to C_1 can be calculated from the output of the navigation system. A plane normal to the vehicle direction at C_2, at a distance H_d in front of the vehicle is taken. The edge segments $a_1 b_1, b_1 c_1, e_1 d_1, \ldots,$ from the image at position C_1 are projected one by one to the ground plane, giving world edge segments AB, BC, CD, . . . , until a projection segment (sueh as CD in the figure) is found to cut the plane. As soon as an interseetion I is found, its image point i is obtained in the image plane of C_2. If this image point i does not fall inside the reduced reetangle of the image, the method is stopped and method A, described above, is initiated to find a replaeement anehor point. If, on the other hand, the image point i falls inside the reduced reetangle, it is ehosen as an anehor point of the first window. This process is repeated for the images of both edges at position C_1.

3. RULE-BASED VISUAL NAVIGATION

The scene model is the central component in the ALV vision system and represents the vehicle's interpretation of its environment. Construetion of the seene model is complex, requiring the direction of vehicle sensors towards objects in the world, the fusion of data from different sensors, and the seleetion of algorithms for image analysis. Methods for performing these tasks are continually evolving as the ALV road following task becomes better understood. Thus, the control structure of a system for constructing a scene model must be flexible enough to accommodate new strategies for performing these tasks. This section describes a new approach to scene model construction which satisfies this requirement and offers other advantages as well.

The task of building a scene model for the ALV involves two major subtasks: 1) deciding what objects to look for and where, and 2) verifying that the objects exist in the world. These two functions are performed by the Scene Model Planner and Scene Model Verifier, respectively; together, they form the Scene Model Builder. The data flow diagram for the Scene Model Builder is presented in Figure 11. The scope of the current research is the design and implementation of the Scene Model Verifier; the Scene Model Planner will be simulated by a human operator. Thus, the Scene Model Verifier will be passed an object hypothesis and will return the object along with a measure of its verifiability. If the object could not be verified, this confidence level will be very low.

Objects in the world exhibit the following relationships:

- *Component Relationships.* For example, an intersection is made up of four connecting roads, stop lights, etc. These component objects, in turn, can be decomposed into their component subobjects, e.g. a road is made up of two road edges, two shoulder edges, and a centerline. However, primitive objects such as lines and surfaces extracted from an image cannot be decomposed.
- *Spatial Relationships.* For example, telephone poles are located near the road and run parallel to the road.
- *Property Inheritance.* For example, although the door to a house and the door to an automobile may be different objects, they both share the generic properties of a door, e.g. handle, dimensions, and direction of opening.

To accommodate these three relationships, frames have been chosen to model objects. A frame is a data structure which encapsulates the relevant knowledge about to an object. This knowledge is stored in a set of slots (or attributes) assigned to the frame. Slots may contain values, e.g. the width of a road patch, or pointers to other frames, e.g. component and spatially related frames. Property inheritance among frames is accomplished by supplementing a frame's slots with the inherited frame's slots. Figure 12 presents an example of two frames representing road patch and planar ribbon objects. The road patch frame inherits the slots of a planar ribbon frame. Through this inheritance, the road patch frame includes slots pointing to left and right segment component frames as well as front and back connected planar ribbon frames. In this example, the road patch is considered to be a specialization of a planar ribbon.

In the context of the scene model builder, a hypothesis is a frame whose slots have not been completely filled. Verifying an object hypothesis, i.e. accumulating evidence supporting the object's presence in the world, amounts to filling in the slots of the object's frame with meaningful values. The mechanism for filling in these slots is provided by a blackboard.

The blackboard can be viewed as a small production system whose facts point to object frames, and whose rule-base contains the rules by which frame slots are filled. To improve efficiency in rule firing and to promote modularity, a separate blackboard exists for each class of object. As with frames, the blackboards exhibit component, spatial, and inheritance relationships.

When an object's class and location can be predicted with sufficient confidence, top-down hypothesis generation is invoked to verify the prediction. If the object can be sufficiently verified, the scene model planner adds it to the scene model. Figure 13 presents an example of top-down hypothesis generation used to verify a road patch hypothesis. To begin the process, a road patch frame is instantiated, some of its slots are filled in, and it is placed on the road patch blackboard. When a new object appears on the blackboard, rules responding to the empty slots invoke knowledge sources to fill in the slots. These activated knowledge sources have access to knowledge contained in the object's frame and any frame directly or indirectly connected to the object's frame. When an empty slot represents a relationship to another frame, e.g. component, a new frame is instantiated on the blackboard representing the class of the related frame. Again, rules fire automatically to fill in the empty slots of the new frame. Eventually, the filling of slots of a primitive object frame requires the extraction of features from an image.

In top-down hypothesis generation, hypotheses deposited on upper blackboards generate hypotheses on lower blackboards. The last slot in each frame to be filled is the confidence of the frame. It is computed based on the values of the other slots as well as values of slots in connected frames. After hypotheses are pushed down the blackboard hierarchy, the confidences are bubbled up the hierarchy. When the confidence of the initially instantiated object is determined, the process terminates. All that remains on the blackboards are the verified objects.

When the scene model planner has little or no knowledge about the vehicle's environment, bottom-up hypothesis generation may be invoked to locate any objects of interest in the image. To begin the process, primitive features are extracted from the image; for each feature, a road patch camera segment frame is deposited on its blackboard. Since the "part-of" slots of these frames are unfilled, the rules generate hypotheses representing the possible objects of which these primitive frames may be a component. The process continues, pushing hypotheses up the hierarchy. In Figure 14, four hypotheses appear on the road patch blackboard. It is up to the scene model planner to decide whether these objects meet the necessary criteria for insertion into the scene model.

The Scene Model Verifier is object oriented and runs in the Franz Lisp environment; both the frames and the blackboards are implemented using Franz Flavors. In addition to slots referencing external entities such as the road map and the scene model, each blackboard object has its own working memory and rule memory. Rules may invoke Lisp functions which, in turn, may invoke C functions to fill a frame's slots.

4. CONCLUSIONS

This report has focused on the design of the Maryland vision system for autonomous navigation of roads and road networks. As mentioned in the Introduction, the ALV project at Maryland also involves support of Martin Marietta, a program of experimentation both in the laboratory and on the Martin Marietta vehicle, and research on parallel algorithms for vision using parallel computers such as the Butterfly and WARP. More information on those aspects of the project are included in Davis et al.[1]

References

[1] Davis, L.S., Chandran, S., Dementhon, D., Gajulapalli, R., Kushner, T.R., Le Moigne, J., Puri, S., and Veatch, P., "The Maryland Visual Navigation System: Status Report," University of Maryland Center for Automation Research Technical Report (in preparation).

[2] Dementhon, D., "A Simulator for Autonomous Navigation of Roads," University of Maryland Center for Automation Research Technical Report (in preparation).

[3] Waxman, A.M., and Sinha, S., "Dynamic Stereo: Passive Ranging to Moving Objects from Relative Image Flows," University of Maryland Center for Automation Research Technical Report 74, July 1984.

[4] Veatch, P., and Davis, L.S., "Obstacle Detection on Road Surfaces," University of Maryland Center for Automation Research Technical Report (in preparation).

[5] Waxman, A.M., Le Moigne, J., Davis, L.S., Liang, E. and Siddalingaiah, T., "A Visual Navigation System for Autonomous Land Vehicles," University of Maryland Center for Automation Research Technical Report 139, July 1985.

[6] Davis, L.S., Kushner, T.R., Le Moigne, J.J., and Waxman, A.M., "Road boundary detection for autonomous vehicle navigation," *Optical Engineering,* Vol. 25, No. 3, pp. 409–414, March 1986.

[7] Dementhon, D., "Inverse Perspective of a Road from a Single Image," University of Maryland Center for Automation Research Technical Report 210, July 1986.

Table 1

COMPARISON OF OBSTACLE DETECTION ALGORITHMS
FOR SENSITIVITY TO SCANNER PERTURBATIONS

SOURCE OF ERRORS:	MAGNITUDE OF ERRORS: (1)		
	RANGE DIFFERENCE	HORIZONTAL DERIVATIVE	VERTICAL DERIVATIVE

3° IN HORIZONTAL ANGLE	19.7(2)
	6.2(3)
		0.3
		0.1
			1.4
			0.1

3° IN ROLL ANGLE	264.6 18.1	6.0 0.3	50.8 0.8
3° IN VERTICAL ANGLE	354.7 20.4	1.6 0.1	74.3 1.0
3° IN EACH ANGLE	1,717.6 38.3	51.4 0.4	790.5 2.1

(1) The absolute values of the expected range (or derivative) at a pixel from an unperturbed scanner minus the expected range (or derivative) from a scanner that has been rotated in the manner listed in the 'source of errors' column.

(2) Maximum absolute error in the entire image.

(3) Maximum absolute error in the center row of the image.

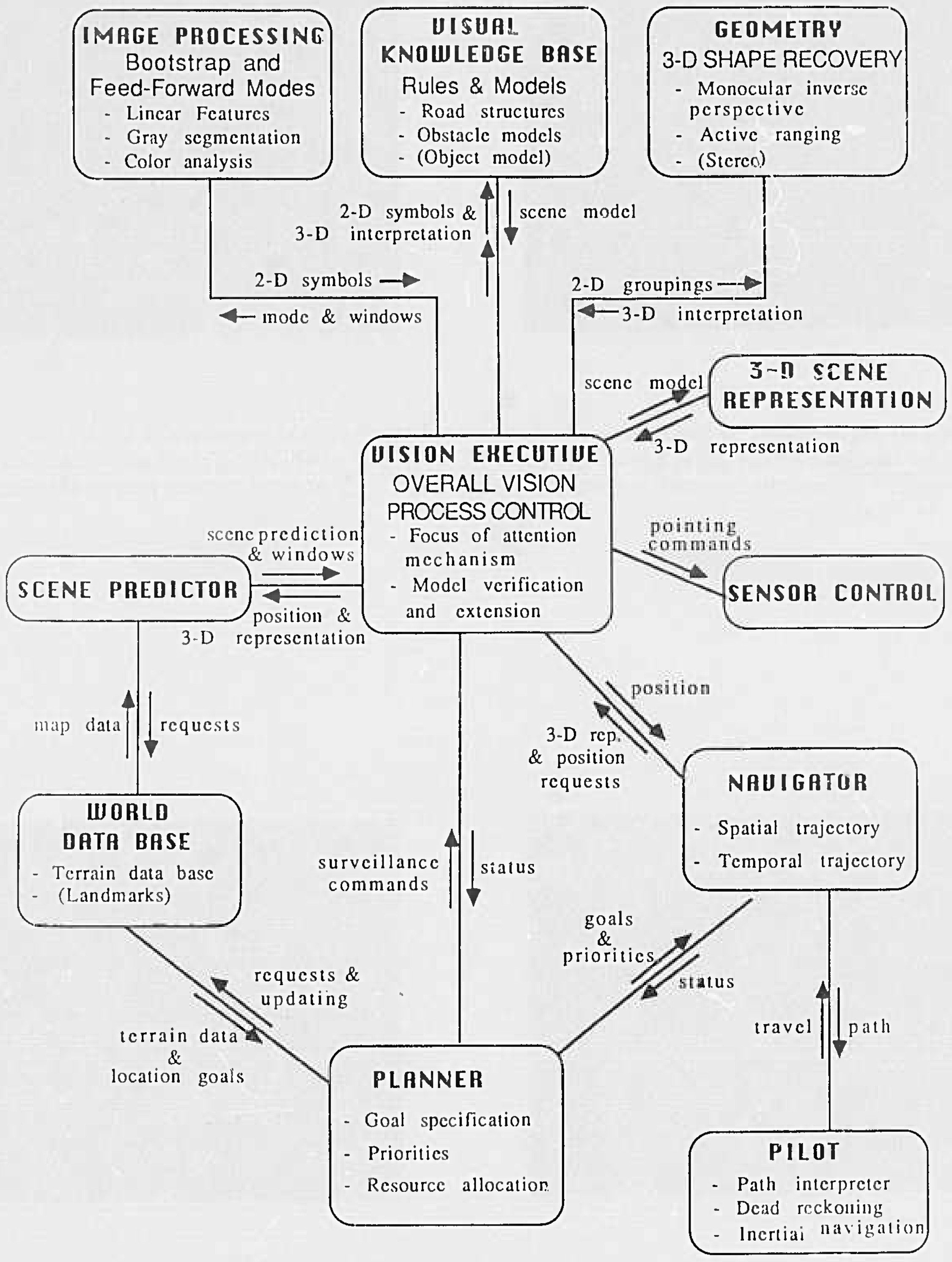


Figure 1: System block diagram

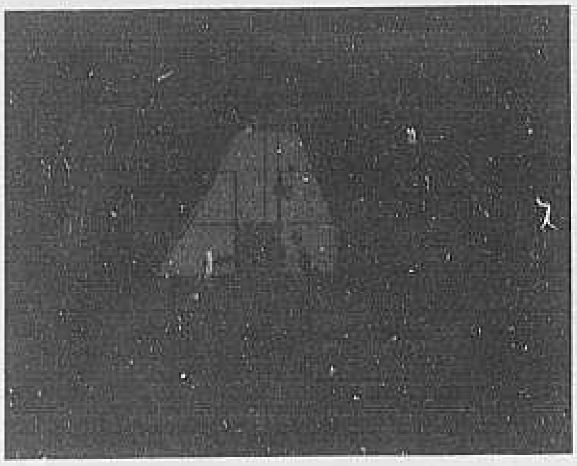


Figure 2: Line segments fit to the border of thresholded image using accumulated angles of road points with respect to window pivots

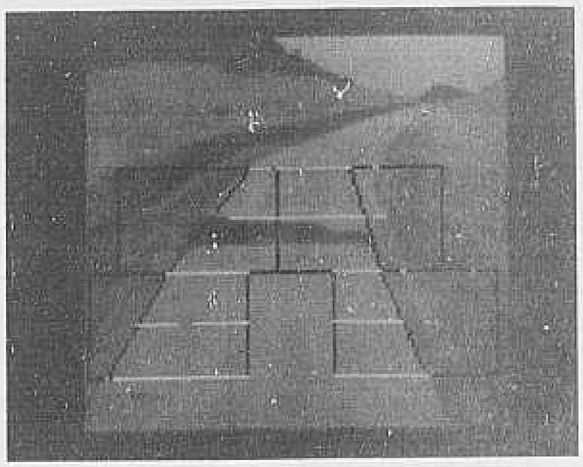


Figure 4: Line segments fit to the border of image exhibiting contrast reversal thresholded around sample grey-scale mean

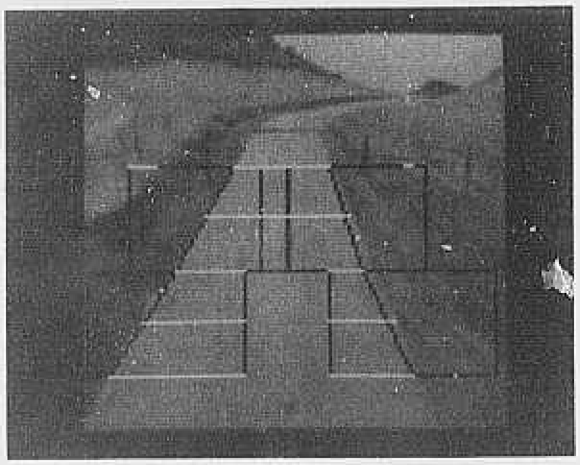


Figure 3: Line segments fit to the border of image using simple threshold determined by minimum-error-criterion

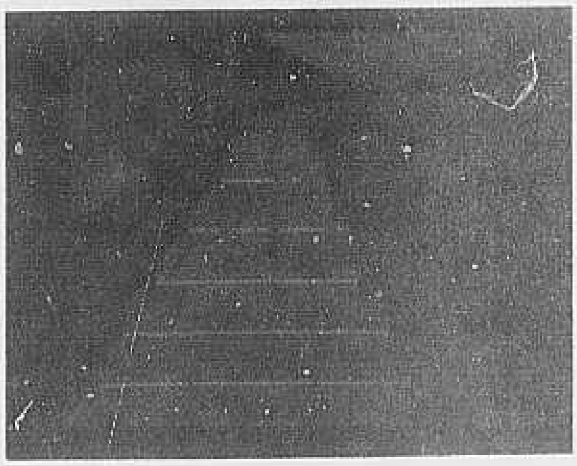
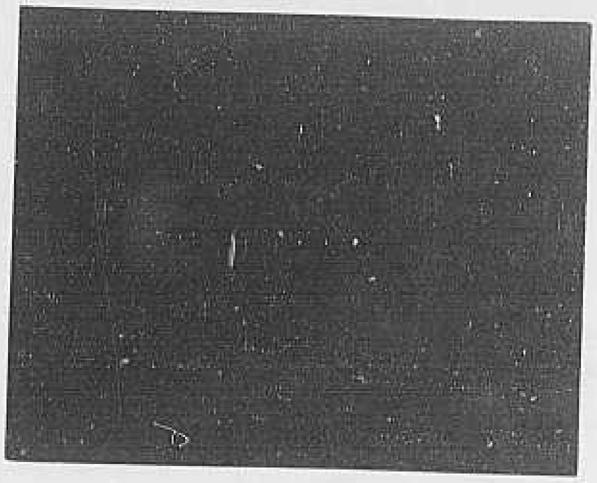
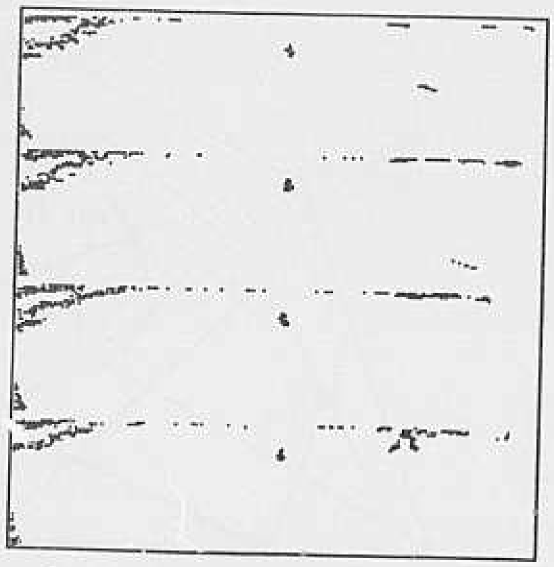


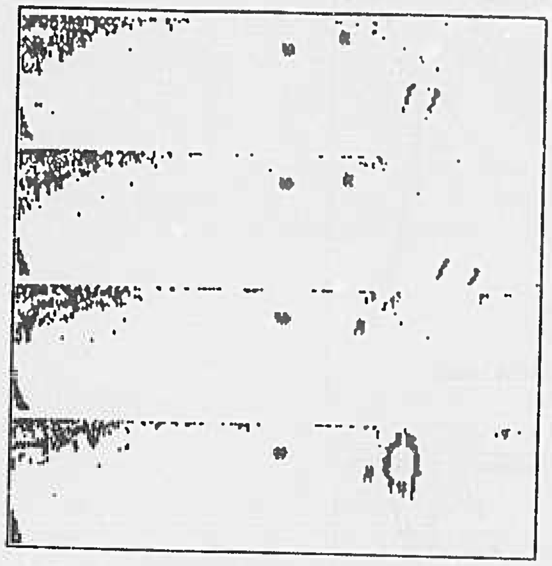
Figure 5: Line segments fit to the border of image using multi-resolution algorithm on thresholded image



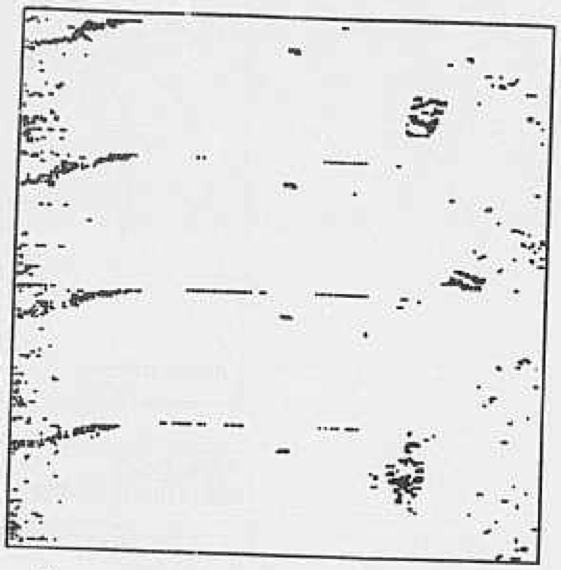
a)



c) THRESHOLDED PHI DERIVATIVES



b) THRESHOLDED THETA DERIVATIVES



d) THRESHOLDED POSITIVE PHI DERIVATIVES

Figure 6: Four range images taken from ERIM range scanner on Martin Martietta test site

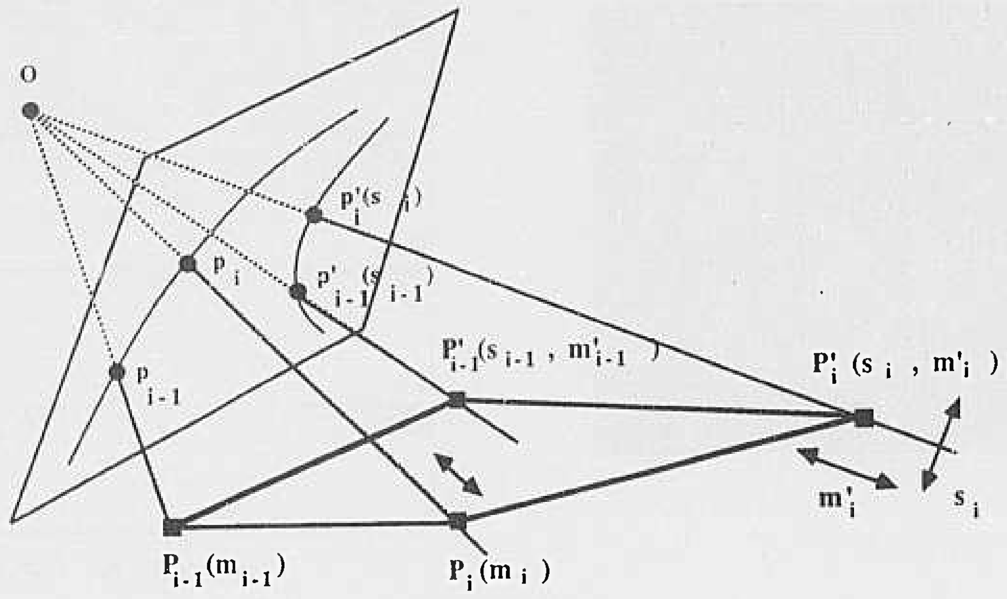
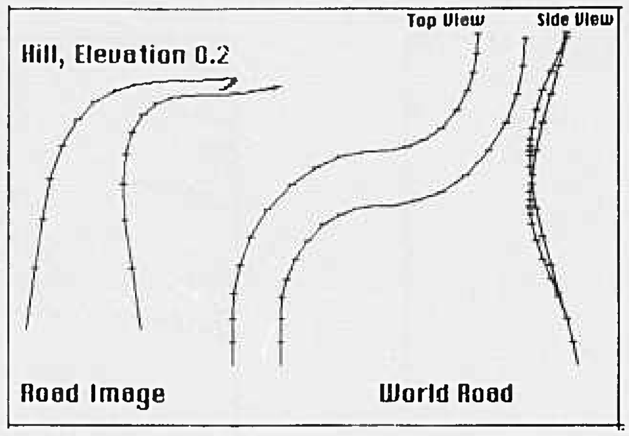


Figure 7: Method for the reconstruction of a road from an image



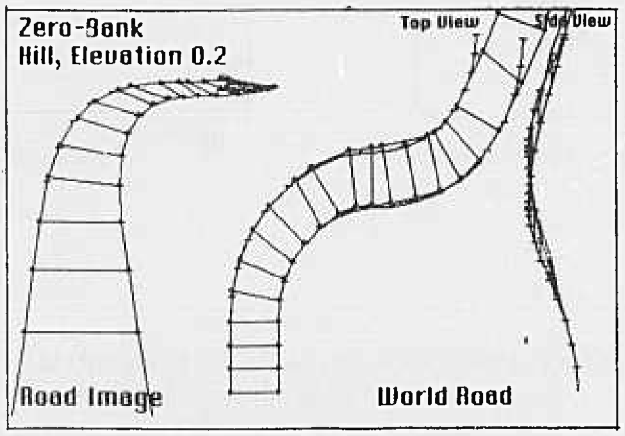


Figure 8: Zero-bank algorithm applied to a synthetic road image

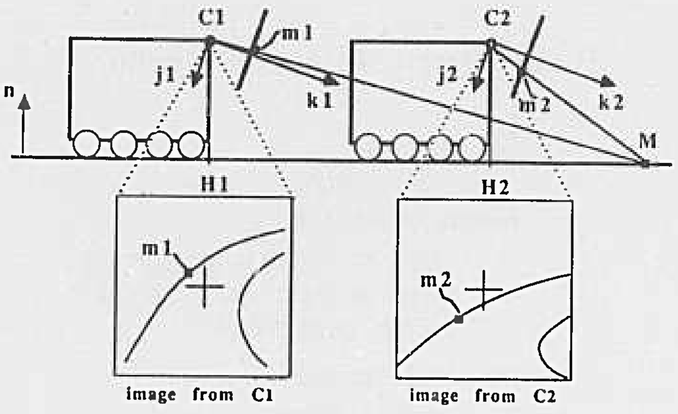


Figure 9: Basic principle of prediction with flat earth assumption

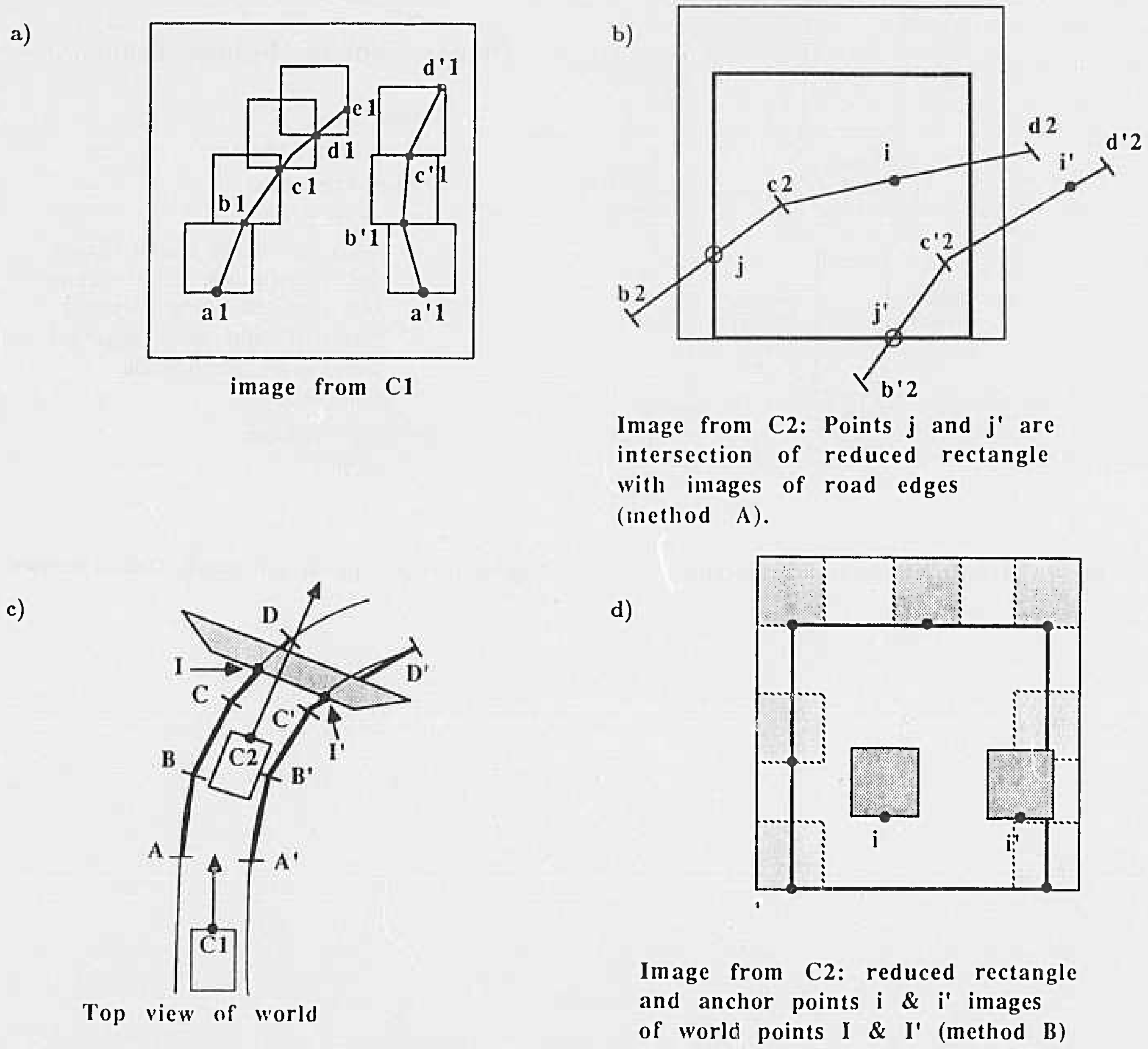

a)
d'1
e1
d1
c'1
c1
b1
b'1
a1
a'1
image from C1
b)
d2
d'2
i
i'
c2
c'2
j
b2
j'
b'2
Image from C2: Points j and j' are intersection of reduced rectangle with images of road edges (method A).
c)
D
I
D'
C
C'
C2
I'
B
B'
A
A'
C1
Top view of world
d)
i
i'
Image from C2: reduced rectangle and anchor points i & i' images of world points I & I' (method B)

Figure 10: Road edge finding methods on thresholded images

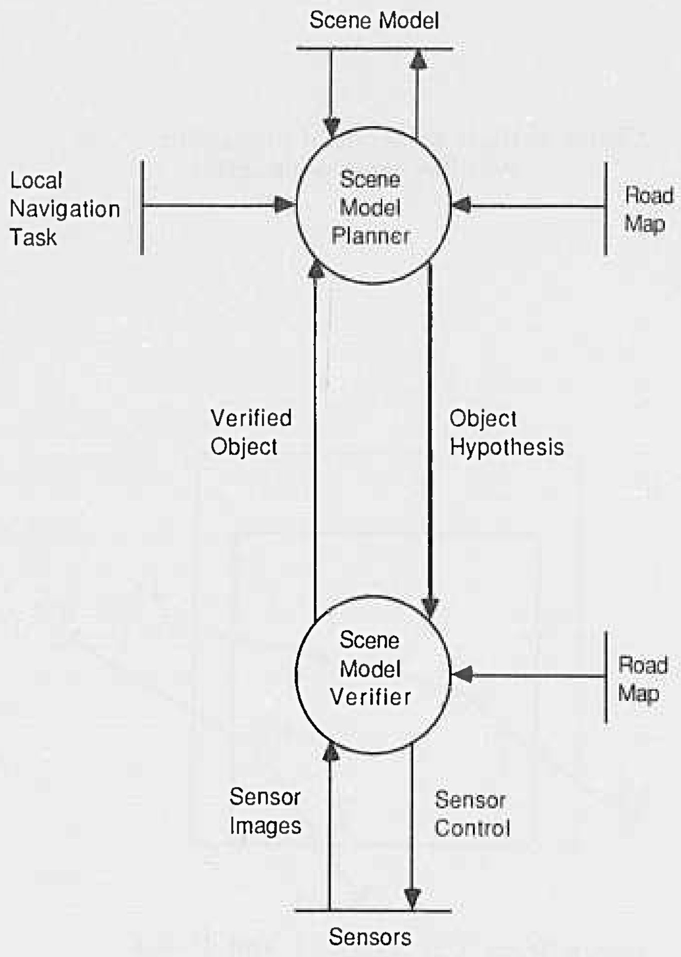


Figure 11: Scene model builder dataflow

Road Patch Frame Definition

Frame Class:
- road patch

Frame Attributes:
- type of left world segment
- type of right world segment
- width confidence

Inherited Frames:
- planar ribbon

Planar Ribbon Frame Definition

Frame Class:
- planar ribbon

Frame Attributes:
- search window
- back connected planar ribbon
- front connected planar ribbon
- has part left world segment
- has part right world segment
- parallelism confidence
- confidence

Inherited Frames:
- none

Figure 12: Road patch and planar ribbon frames

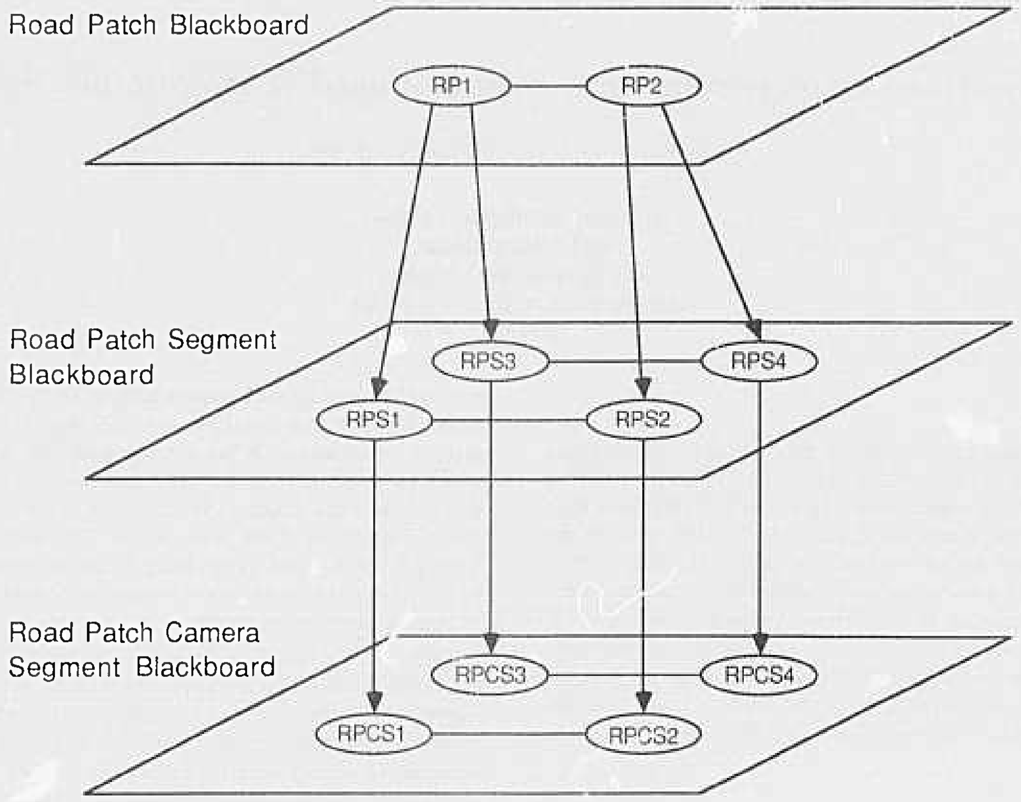

Road Patch Blackboard
RP1
RP2
Road Patch Segment Blackboard
RPS3
RPS4
RPS1
RPS2
Road Patch Camera Segment Blackboard
RPCS3
RPCS4
RPCS1
RPCS2

Figure 13: Top-down hypothesis generation

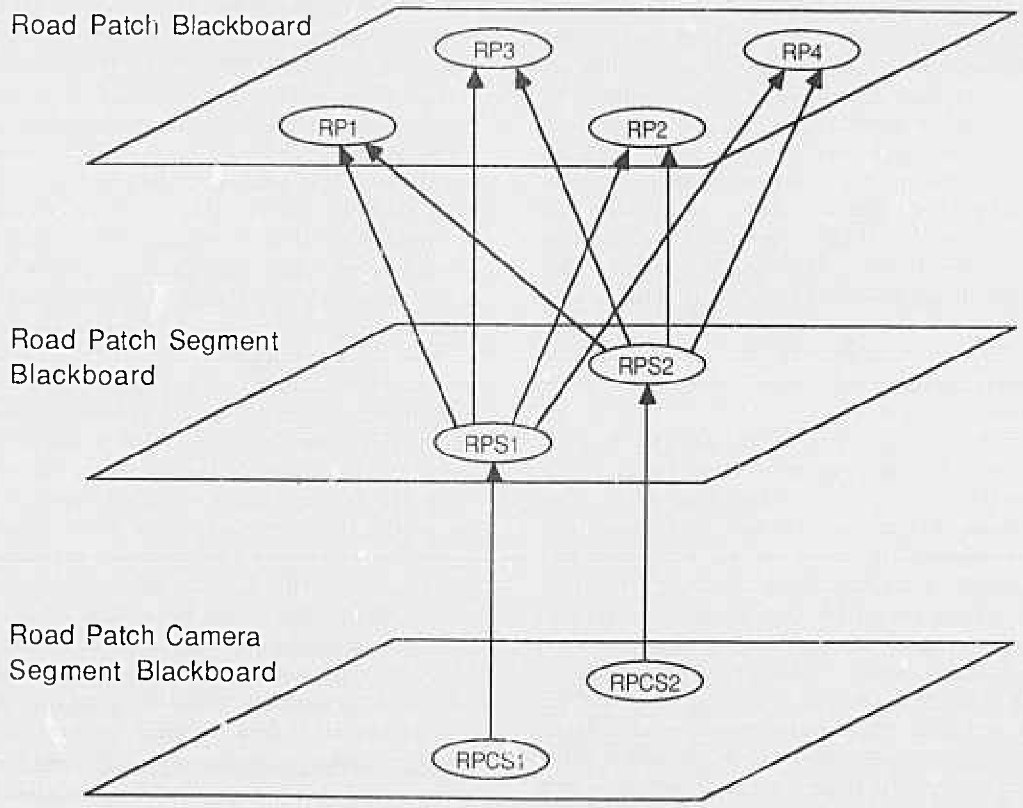

Road Patch Blackboard
RP3
RP4
RP1
RP2
Road Patch Segment Blackboard
RPS2
RPS1
Road Patch Camera Segment Blackboard
RPCS2
RPCS1

Figure 14: Bottom-up hypothesis generation

Information Management in a Sensor-based Autonomous System

Grahame B. Smith and Thomas M. Strat

Artificial Intelligence Center
SRI International
333 Ravenswood Avenue
Menlo Park, California 94025

Abstract

This paper describes an information manager that is at the core of a sensor-based autonomous system. The manager provides the means by which sensor-based data can be integrated with stored knowledge to construct a world model that encodes the information needed for autonomous behavior. The autonomous system consists of a community of independent processes, each responsible for a portion of the system's goals, and each of which interacts with an active database to share results and to achieve goals that require the combined effort of more than one process. The information manager facilitates the sharing of data and provides an organization needed to accomplish system-level goals.

1 Introduction

This paper describes a knowledge system that integrates data from a variety of sensors, as well as from other sources of stored knowledge, to form a world model that is used by an autonomous system to plan and execute its tasks. The overall architecture of our knowledge system can be viewed as a *community* of interacting *processes*, each of which has its own limited goals and expertise, but all of which cooperate to achieve the higher goals of the system. The various processes may represent sensors, interpreters, controllers, user-interface drivers, planners, or any other information processor that can be imagined. Each process can be both a producer of information and a consumer. Information is shared among processes by allowing them to read data stored by other processes and to update that information. Each process continually and asynchronously updates information based on sensor readings, deductions, renderings, or other interpretations that it makes.

A simple example of object recognition may help to show the system approach. During cross-country navigation an autonomous land vehicle must be able to detect small gullies in its path. Let us suppose that the vehicle has a range sensor and analyser that can segment the world into surface terrain and objects on that terrain, a drainage expert that can derive run-off patterns from surface topography, a multi-spectral analyser that can classify objects on the basis of their spectral signature, a gully detector that determines the likelihood of a gully in the vehicle's path on the basis of the surface drainage pattern and the vegetation cover, and a pilot that makes navigational decision. Each process contributes expert knowledge about aspects of the world. The determination that there is a gully is not made by a single process that calculates drainage pattern, vegetation type, etc., but rather, by each expert looking at the objects that have been placed in the database and deciding if it can contribute further information. When there is sufficient information from which to conclude that there is a gully present, the gully expert will annotate the database to this effect. This will then influence steerage decisions of the pilot process. The decision that there is a gully is not reached by applying a fixed sequence of operations to the data but is made when the available evidence is sufficient to support such a conclusion.

Each process is a *knowledge source* that brings its expertise to the processing of the data that represent the known state of the world. These processes span the range from low-level image processing to symbolic manipulation, and their output will be available for use by all other knowledge sources. The system design is one of expert knowledge sources that know how to draw conclusions based on the available data but whose knowledge does not necessarily define the actions required to obtain that data. Such procedures are run independently of the processes that use their conclusions. The database that stores these conclusions must be able to accept a vast assortment of data types and make them available to requesting processes.

Our system includes a global database through which information is shared. Because all processes share information, the communication bandwidth between this database and the various processes is of concern. If the granularity of the information to be shared is too fine, then the communication channels will be overloaded with an enormous number of transactions, each of which involves small amounts of data, while a granularity that is too coarse requires complex knowledge sources that are beyond our ability to construct. We view the knowledge sources as substantial entities that attempt to share data objects that are composite in nature. For example, we do not expect that an image-processing routine would write intensity-edge information into the database, but rather that it would share conclusions about the physical objects that are in the world. Of course, these objects may not be identified, nor need they be the final partitioning of the scene into world objects, but they will be entities with which other processes can associate parameters and semantics. This does not mean that the database contains only symbolic objects, but rather that it contains objects that have some semantic character, such as a horizontal planar surface with approximately constant albedo. This minimizes the volume of transactions within the system by associating a significant amount of data with each transaction.

Some knowledge sources may need to communicate with others at a level that is not provided by the global database. Such communication is private to those sources, and implementation is the responsibilty of the designers of those processes. This level of information sharing often entails a certain computational

The work reported here was supported by the Defense Advanced Research Projects Agency under Contract DACA76-85-C-0004.

speed requirement and usually a processing sequence that can be prespecified. Although any pair of processes may use this form of close coupling, we have focused on expediting the sharing of information that is of a higher level and is substantially unstructured.

If processes are to communicate through the database, the language of communication must be rich enough to allow items to be shared. Relevant information extracted from the database is of little use if the receiving process cannot understand it or make use of it. With the diversity of information that is available, we choose to share that information through *semantic labels* that classify the information in the database. These labels must reflect the multiple levels of specificity inherent in the information itself. The labels form a *vocabulary* describing the information that is stored in the database. Accessing information by means of the semantic label allows processes to be independent of the particular data syntax used to store the information. We allow database access through logical combinations of the semantic labels, as well as through procedural definitions passed to the database so that a user may supplement the vocabulary with additional terms. Passing procedural definitions to the database also reduces the communication bandwidth otherwise needed to return the results.

A system that views processes as individual experts that may make conflicting interpretations of the data must have a policy to determine what is stored in the database. For example, if two processes determine the height of a particular tree to be substantially different, whose opinion should be stored: the last one given, that of the process with more expertise, or the average of the two? There is no "correct" way to determine a single value. Usually, information integration is accomplished as the data are inserted into the database, and the data that are then retained are expected to be conflict-free. In our system, all processes are considered equal, and only their *opinions* are stored. This approach reflects the view that conclusions are not only a function of the data used, but also of the knowledge sources that provide that data, and the anticipated use of the conclusion. The user of information should have the opportunity to filter that information with knowledge of both its content and its source. Information in the data store can be modified only by the process that created it, although other processes can cast their opinions.

Any system that consists of a collection of independent, asynchronous processes must have a control mechanism that coordinates these processes to achieve the system's goals. In our design, each process is continually active, going about its task of processing the data that define the current state of the world and placing its opinions in the database. When certain combinations of data occur, we must be able to interrupt particular processes and have them deal with this new information. In our previous example of cross-country navigation we needed to detect gullies. A clump of snowberry bushes is often an indication of a gully. Consequently, we should activate the gully detector whenever snowberries are detected. A daemon approach is used to implement this strategy. Daemons are placed in the database by the processes that should be informed when particular events occur, and the processes are responsible for determining how to proceed when they are interrupted by these daemons. Control by means of the database is therefore data driven. Alternatively, any process is free to call procedures that are imbedded within another process, thus allowing control to be passed by procedure call.

Control that is data driven is unlikely to be coordinated to achieve the goals of the system if those goals are not available to the various processes that are performing the data transformations. An important part of sensory integration is planning which activities will contribute to the more general goals of the larger system in which the sensory system is embedded. In our case, we interface with the goals of a *planning system* that controls the activities of an autonomous vehicle. A planning system is viewed simply as another process or set of processes that may access the database. The list of tasks that the vision system is attemping to achieve serves as data that individual processes must use to prioritize their own activities. Conclusions and data transformations, no matter how correct or clever, are irrelevant if they are unrelated to fulfilling the mission of the highest-level system.

2 Database

The database that we have designed to store the domain data has many of the usual database features. It stores a collection of *data tokens* that contain the domain information and has a set of indexing structures overlaid on these tokens so that data manipulations based on the domain requirements, such as data retrieval, may be implemented efficiently. Unlike many vision-system databases, the database has a continuity of life that exceeds a single execution of the system. In this respect it is much more like a conventional database, whose integrity and usefulness must persist over an extended period. Data acquired during execution of the system becomes knowledge stored in the database for future use.

To ensure that the internal integrity of the database is maintained, processes do not have direct access to the data tokens; instead copies of the data are transferred between the database and the process. Clearly, data copying is computationally expensive, which is incompatible with real-time performance. For this reason, a mechanism is provided in the data access language that allows a process to pass a trustable procedure to the database so that internal processing can be used to minimize the volume of data transferred and the amount of copying that is necessary. This approach for controlling integrity is dictated by a development environment in which the system is not built by a single person or group but rather is a set of processes provided by disparate implementors. Protecting the data from corruption by an errant process is necessary if we are to avoid rolling back the database to a previous version or editing it between actual uses. However the mechanism used to reduce data copying, sometimes at the expense of jeopardizing integrity, is desirable for certain time-critical processes so that they may achieve real-time performance.

Because the opinions of all processes are stored, the database will contain conflicting and incompatible views of the state of the world. Some processes may exist solely for the purpose of resolving such data inconsistencies. Of course, even these processes will only be allowed to cast an opinion. User processes may choose to take more notice of the opinions of these conflict resolution processes than of the opinions of processes whose conclusions are drawn from less data. The conflict resolution processes will continually process data in the database (as spare computational resources allow), but they are conservative in nature, preferring not to cast an opinion unless they have overwhelming evidence to support their conclusion. However, a user process may call one of these conflict resolvers to cast an opinion even if it would not have otherwise intervened. Our approach then is to allow inconsistenc[illegible] o be resolved whenever the data is sufficient to support the resolution, or whenever a user process requires that resolution, i.e. at access time. This approach differs from other approaches that attempt to maintain a consistent data set: in these approaches resolution must occur at insertion time. The approach we adopt is to resolve when necessary, rather than to resolve always. Often a decision-making process can take action without the need to expend resources in resolving data discrepancies. For example, the navigation module of an autonomous land vehicle may be faced with the conflicting data that the object ahead is either a tree or a telephone pole. If the task is to move forward avoiding obstacles, the vision system does not need to resolve whether the object ahead is a tree or telephone pole. The resolution requirement is a function of the task, not simply the data.

A database that stores opinions will rapidly consume storage resources unless a mechanism is provided that will allow data to be deleted (or at least archived). A process that is the supplier of data may have little ability to evaluate the usefulness of that data, yet it is the useful data that we would want available in the database. The approach adopted is to have processes sponsor data; that is, a process (probably a process that uses a particular data token) will allow that data token to be "charged" against its resource allocation. Many processes can sponsor a single data token, and they are charged proportionately. When a process nears its resource limit (or at any time) it can withdraw its sponsorship of any data that it has sponsored. Data that are unsponsored are available for garbage collection (these data may be archived or deleted). In this manner each process is responsible for deciding what data it finds useful, and this collection of data forms the base of current available information. Clearly, this procedure is not fail safe. Critical data may be removed before their criticality is realized. However, the criticality of data is measured in terms of a process's willingness to pay for it and presumably in terms of the current usefulness of that data.

An unsponsored data token will not necessarily be removed immediately. An information producer may not wish to sponsor data for which it has little use, so it may be some time before a sponsor for this information is found. To avoid deleting useful data, the process whose job is to remove data tokens evaluates additional information, such as length of time the token has been in the database, as well as sponsorship information, before it is removed. Data removal is a continuous process, so that the database can be assured of having adequate storage when time-critical tasks demand that computational resources for garbage collection be suspended.

Each process in the system does not have the same resource allocation. At particular times some processes may be more valuable than others. For example, the gully detector used by an autonomous vehicle is a vital process during cross-country navigation but is rarely used during road traversal. One process has the task of allocating database resources to the processes performing useful tasks. The allocation is based on the relative importance of the task and on the frequency with which data tokens are consumed by the process performing that task. Such a frequency measure is a moving statistic that allows the allocation to adapt to the current situation. Data tokens are time stamped to indicate the last time they were modified — that is, the last time a new opinion was added to one of the data slots — and they are time stamped for last use. The time stamps provide data for the resource allocator and the garbage collector.

Data tokens are produced by individual processes and are passed to the database for storage and subsequent retrieval. For the database to access information from within the token, or for a requesting process to be able to extract information from a token, each must either know the form of that information or have some procedure for recovering it. In the design of a system one can use a standard structure for a data token, such as a record structure in which the position of parameter slots are known, or a standard syntax for the token can be used, such as a list of attribute-value pairs, or one can add functions that retrieve values from the internal data structure of the token, i.e., procedural attachment.

With standard structures, position, rather than name, gives access to the data but all processes are required to use some predetermined set of structures. In a system in which different processes do entirely different tasks, it is unlikely that one could find, no matter how clever, a small number of data structures that would be natural representations of all the data for all the processes that use that data.

With both fixed syntax and procedural attachment to a data structure, a vocabulary of terms is needed to access the data slots. This is the approach we take. A vocabulary of terms is used that spans the entities and relationships of interest in the application domain. For an autonomous land vehicle, the vocabulary consists of words or labels that describe the outdoor environment, e.g. tree and height, so a process could ask a data token that represents a tree for that tree's height. The actual structure used to hold the data can be invisible to the process which gains access to the information through the labels. The labels must be known by all processes that wish to access this information in the database. This semantic level does seem to be the appropriate level on which to share information.

Should one use a fixed syntax like attribute-value pairs to hold the information in the database and provide a simple routine to retrieve the value given the attribute, or should one use the more complex approach of attaching to a data structure a set of functions that can retrieve the value of a data slot given the slot name? We take the latter approach to increase the functionality that is available when a value is retrieved based on slot name. From the point of view of systems building, in which parts of the system are built by independent groups, this approach places the decisions for the form of the data structure and the accessing functionality within a single group and provides a clean interface with the database. Each process can select its own internal representations for the data it produces, and that data can be shared through access functions that are based on terms or labels in the vocabulary which describes the underlying domain. A common vocabulary requires that each process know how to translate from its internal representation to information in vocabulary form. This avoids the need for each process to know how to translate into the individual representations used by other processes. Additionally, new processes can be added to the system without retrofitting the new representations to the older processes.

A collection of data tokens is not a database unless there is a means of accessing the information in the collection in a manner that does not require a search through the entire set. A set of indexing structures that allows access in a more direct manner must be based on the subsets of the data that need to be retrieved. These structures are therefore based on the domain requirements and relate to the semantics of the actual data stored. Our architecture for sensory integration is implemented in the task domain of an autonomous land vehicle navigation. The indexing structures that we use are associated with the need to retrieve information that is appropriately grouped for the task

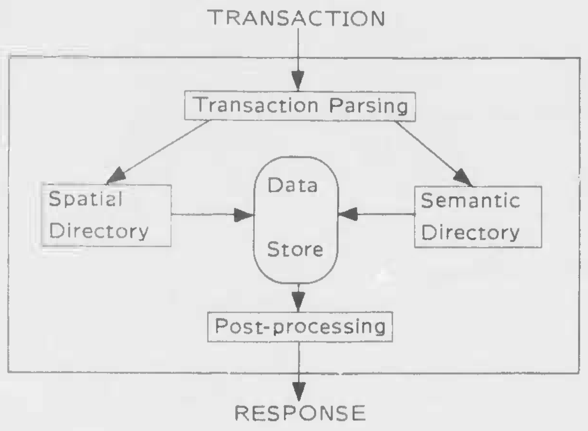


Figure 1: Transaction Processing — Autonomous Land Vehicle Database. Access to the data store is by means of a spatial directory, a semantic directory, or both.

of navigation in the three-dimensional world. A *spatial directory* that forms subsets of the data based on spatial location, and a *semantic directory* that forms subsets of the data based on object class are the principle indexing schemes that we use to organize storage and retrieval of data tokens. Figure 1 gives an overview of transaction processing by means of directories in the database designed to support autonomous vehicle navigation.

3 Spatial Directory

The spatial directory organizes the data tokens into groups determined by spatial location. Because an autonomous vehicle may roam about in an extensive environment, we need a representation of that environment that can deal with its spatial extent. In addition, the representation must be efficient in indexing data when the data are distributed nonuniformly over the environment. Data will need to be accessed at various levels of resolution depending on the task that is being addressed. Route planning needs lower-resolution data than does, for example, landmark identification or obstacle avoidance. The world is three-dimensional but the vehicle is restricted to a two-dimensional surface embedded in this world. Although there are many reasons for choosing a two-dimensional index, such as latitude and longitude, and then representing the third dimension as a data value, we chose to use a three-dimensional index. Our selection was motivated by the advantage such an index gives in encoding spatial relations within the directory, in generating visibility information, and in using this architecture in other spatial domains in which movement is not restricted to a two-dimensional surface.

The three-dimensional index selects a volume in space that we represent as *voxels* [4]. The largest voxel is the world, which is subdivided into smaller volumes as we need to represent spatial position with higher precision. The index granularity is fine enough to be able to position an object in a volume that is precise enough for the application. Recall that this index is an index into a directory; in the directory cell are pointers to the data tokens associated with the volume of space represented by this index. Data tokens need not be placed in the directory at the finest index available but only at the precision with which their spatial location is known. A tree whose position is unknown would be placed in the largest voxel; this being the voxel that represents the entire world.

The voxel-based directory not only gives a range of position resolutions, it also allows different parts of the world to use different resolutions for storing data. In parts of the world that have little data associated with them one may choose to place all the pointers to data tokens representing objects in coarse-grained volumes, while the part of the world in which the vehicle is active can be subdivided into finely partitioned volumes. This not only provides for multiple resolution access, but also allows one to select resolution relevant to the area concerned.

In selecting a voxel-based representation of space, one has the option of dividing that space into regular voxels in which all voxels, at a given level of subdivision of the space, are of equal size, or choosing to divide the space into irregularly sized chunks. Irregularly sized voxels have some attractions, as they allow irregularly shaped objects to be confined, and hence indexed, within a volume that matches them. Uniformly sized voxels often are unnecessarily large when they are large enough to contain an irregularly shaped object. However, if irregularly sized voxels are used multiple indices are needed to allow for overlapping voxels that are indexing different irregularly sized objects in the same volume of space – thus increasing the computational load. We therefore use a regular subdivision of space in which each voxel is subdivided into eight equally sized and shaped smaller voxels.

In making this choice we must address the problem of indexing objects whose shape does not match this partitioning of space. Generally, it is easy to place stationary compact objects within a voxel that can completely contain them, but objects like linear structures, surfaces, and moving objects require alternative approaches. A linear structure like a road, river, telephone wire, or fence is stored as a single data token, but pointers are placed in all the voxels through which the structure passes. The smallest-sized voxels that are appropriate are used; for example, the voxel size for a road will be determined by the road width so that it is assured that the road "fits" within the voxel. The same approach is taken with other extended objects, such as surfaces: a single data token has pointers to it from the set of voxels through which the surface passes.

The size of the voxel is selected by the process inserting the surface into the database, based on such factors as accuracy of the surface shape, and extent. Recall that this placement in space is to aid retrieval, not to specify exactly where things are. Detailed location information in available from within the data token. There is no need to place objects in the spatial directory in the smallest voxel that might be possible.

Moving objects are usually compact so they present little problem in placement at their current position, but there may be occasions when it is desirable to represent their track in the directory. The same approach is used as was adopted for linear structures and extended objects: the moving objects are represented by a single data token that is pointed to by the voxels associated with its track.

An advantage of a multiresolution spatial directory is the ease with which approximate location can be represented. An object is placed in a voxel that is large enough to contain the limits of its possible locations. Object location may be approximate because of image processing errors when detecting objects in imagery, or because of lack of knowledge of a sensor's exact position. The latter is particularly relevant in the case of an autonomous vehicle. Data can be added to the database before its position is known, and then, when better location information becomes available, the directory can be updated by moving the data to

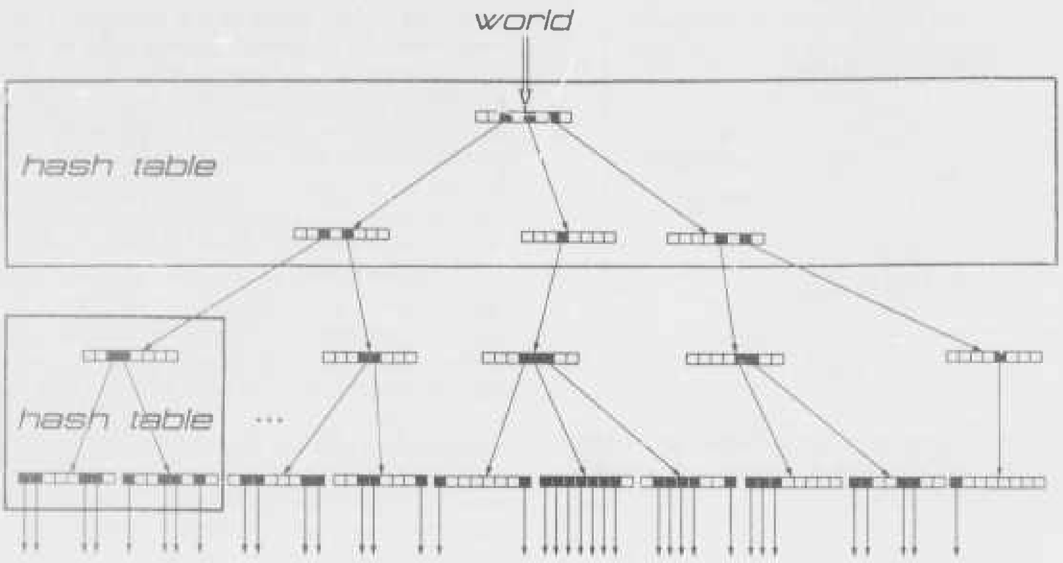


Figure 2: Octree Representation of the Voxel Description of Space. Hash tables are used to implement the octree; more than one level of the octree is stored in a single hash table.

a smaller volume. If this is not done, the data will be retrieved and examined whenever requests are processed for data from the original larger voxel. A background process whose task is to move each object to its most precise location within the directory (when processing resources are available) accomplishes the directory update and thereby achieves retrieval efficiency. Hence all data can be directly inserted into one directory whether their location is known accurately or only approximately.

Having all data, whose position is known or uncertain within one directory structure allows us to respond easily to data retrieval requests that want "all objects that are within a certain volume in space" as well as "all objects that are possibly within that particular volume of space." Clearly, in the task domain of an autonomous land vehicle, knowing what *might be* ahead and what *is* ahead is necessary for competent navigation and obstacle avoidance. For example, a landmark recognition process needs to know what objects are definitely in some volume, while an obstacle avoidance process is interested in all objects that are possibly in front of the vehicle. Within the voxel structure, "within a volume" maps to the tree of voxels below (finer than) the voxel containing the volume while "possibly within a volume" maps to the tree above (coarser than) the voxel containing the volume.

When data can be retrieved on the basis of their location, then retrievals on the basis of spatial relations are also possible. The spatial directory encodes the spatial relationships between items stored in the database. As objects are moved or their spatial positions refined, these spatial relations are maintained without additional processing resources. New objects entered into the database encode their spatial relationship with previously entered data. In our task domain, we expect to retrieve items based on relative position — objects to the right of the road, trees casting shadows on the road, and so on. Having an indexing structure that matches the world structure allows this without the overhead that would be presented by alternative schemes, such as a relational database.

The reduction of computational resources used to maintain the database was also instrumental in our treatment of time. The database is always assumed to represent the world at current time. If historical information is to be stored, then it must be time-stamped, otherwise it is implied that the data reflect the state of the world as it currently is. This approach was adopted so that one could avoid elements of the traditional frame problem [1]: if time is a parameter of the data token, then this token has to be updated even when the real data has not changed but time has passed. We take the usual approach adopted in conventional databases, in that information is assumed to be still true if it has not been altered or specifically marked as applying only to some particular interval of time.

The spatial directory is organized as an *octree* [3] of voxels that span the world. Specifically, a "pointerless" octree that is implemented by multiple hash tables is employed. The use of an octree to implement a voxel representation is natural; the selection of the pointerless approach was based on the expectation that many voxels will contain no data, and many voxels will not be subdivided into smaller units. Hence the more usual approach of using cells with explicit pointers to the finer cells will produce many cells containing mainly null pointers. With the pointerless approach, only voxels that contain data tokens are allocated any storage, and null pointers are not used. Figure 2 shows an abstract view (using null pointers) of the way an octree is used to represent the voxel description of the world. The number of levels of hash tables is in fact somewhat less than the number of octree levels, because several octree levels are stored in a single hash table, as shown in Figure 2.

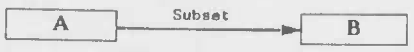
4 Semantic Directory

The spatial directory provides an indexing scheme that matches the spatial nature of the data in the task domain; the semantic directory provides an indexing scheme that matches the semantic nature of the data in that domain. As previously mentioned, a vocabulary of terms is used to facilitate communication between processes. The semantic directory specifies these terms and defines the connections between them. The vocabulary provides a set of labels that is used to describe the data tokens in the database. Such a set is dependent on the task domain, and for autonomous land vehicles we use terms that label objects in the outdoor environment, such as tree, road, rock, meadow, or ditch, as well as terms with less specificity, such as immovable_object, obstacle, or object.

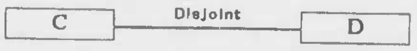
The need for terms that define the semantics of things in the world at various levels of abstraction or multiple levels of resolution is apparent if one wishes to interpret imagery as seen from a moving vehicle: objects usually appear first at a distance, at poor resolution, and gradually change form as one approaches them. The needed levels of abstraction are a function of the available processes and their ability to instantiate the terms. There is no point in being able to describe leaves on a tree if the sensors are incapable of resolving objects that small. Equally there is no point in describing trees as belonging to the superset "wooden objects" if no process makes use of that set. The vocabulary choice that we have made is based on an assessment of the competence of low-level image processing routines and the requirements of higher-level processes. The choice is critical to sensory integration, for within the vocabulary we are restricting the means of integration, the information that higher-level processes can transfer to the low-level routines (and vice versa), and the functionality requirements of both higher and lower-level processes. In absolute terms, successful sensor integration demands selection of an appropriate vocabulary.

Any vocabulary whose constituent terms span a wide range of specificity in a domain will include terms that are related to one another. The second component of the semantic directory, a semantic network [1], defines these connections. The network itself has two parts: one which defines the specialization of terms by a graph, that is, a lattice that specifies *subset/superset* relations and that is augmented by the inclusion of the *disjoint set* relation, and a second part that describes the decomposition of composite objects into parts. While the first part indicates relationships that must hold, such as "a pine tree is a tree," the second decomposes composite objects into parts that are usually present, such as "fire engines usually have ladders." The first part of the network is used for inference; for example, in inferring that a pine_tree is a tree, which is an immovable object, which is an object, and so on. The second part gives default values that may be used to trigger some process to find them, or may be used by an evidential reasoning process that is attempting, say, to classify an object based on what has been detected and what one might expect to see when viewing that particular object. For example, when a process is attempting to decide whether an object that is composed of several vertical rectangular objects and some horizontal lines could be a portion of a fence, knowledge of the expected parts of a fence is crucial to that determination. Additionally, the network provides a means for inheriting properties from a more general class; e.g. if a tree is usually composed of branches, leaves, and a trunk, then a subclass, like pine_trees, will inherit this parts decomposition as its default description. The approach taken reflects on one hand, the need for the system to reason about objects, while, on the other hand, the system must be able to recognize composite objects on the basis of their likely parts. A mechanism for logical inference and a mechanism for object decomposition that is usual but not unequivocal is therefore provided.

The semantic network is implemented as a graph in which the nodes represent the vocabulary items and the labeled arcs represent the relationships among terms. Both the subset/superset and disjoint set relations, together with the composite object decomposition, are combined on the one graph using various labels on the arcs to distinguish between them. For example, the lattice fragment



encodes the sentence $\forall x : (A(x) \rightarrow B(x))$, while



encodes $\neg\exists x : (C(x) \wedge D(x))$. This network representation allows selected inferences to be made rapidly through graph operations. The particular implementation allows display of the semantic network in its entirety or of selected clusters of related information. Figure 3 shows a small part of the semantic network. The graphical display of the network is the interface used to build the semantic directory and to add new words and relations to the vocabulary.

Each node of the semantic network is associated with a vocabulary term, and has pointers to all the data tokens in the database that have been labeled with this term. The nodes of the semantic network can be directly accessed by the vocabulary label, and thus provide a directory to data tokens on the basis of the semantic label. Although we view the semantic directory as a graph structure and display the semantic network as a graph, the implementation uses hash tables for speed of access. When data tokens are added to the database or when additional labels are added to a token's description, the semantic directory is updated appropriately.

Data tokens are attached to the most specific network nodes possible. If, for example, a data token had been labeled by a process as being a paved_road, then it is attached only to the semantic network node for paved_road even though all paved_roads are known to be roadways. This approach was adopted to save storage as well as to provide a straight-forward implementation of the retrieval request to return all objects that are paved_roads as opposed to all objects that might be paved_roads. The second descriptor includes objects in the more general class "roadways" as well as those labeled "paved_roads." Paved_roads are found attached to the nodes of the lattice that form the tree rooted at the node labeled paved_road, whereas roadways that *might be* paved_roads are found attached to the nodes of the network tree *above* the node labeled paved_road. This arrangement parallels the mechanisms used in the spatial directory to find objects that are at a particular location, as opposed to those that might be at that location. It is the responsibility of the access routines to retrieve the appropriate items from the database by means of the semantic network.

The semantic network serves partly as a definition of the meaning of concepts. If a process designer wishes to know what questions he can ask of a data token that is, for example, a tree, the network specifies the relevant terms, such as height, or color. The semantic network defines more than just the communication language between processes; it defines something of the domain concepts that all processes must use. However, while the concept "tree," for example, may be seen in the semantic network to include pine_trees, and oak_trees, and so on, and while a tree

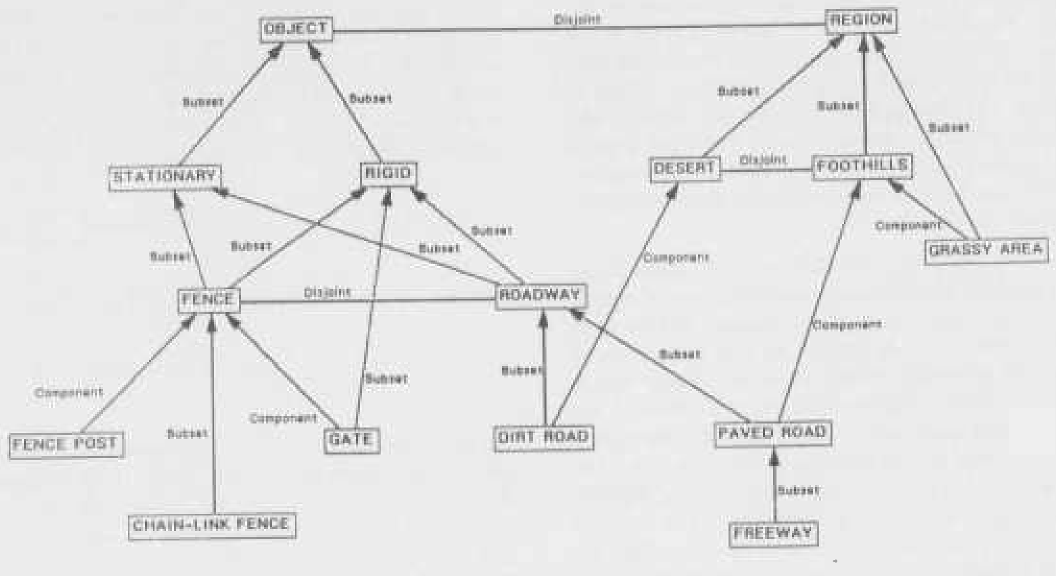


Figure 3: Semantic Directory. Implemented as a semantic network, it gives access to the database data tokens by means of their semantic type.

is an immovable_object and an object, and while it has parts (and properties) of height, and color, it is not "defined" by the network. The network does not define for a process the concept "tree;" it specifies only the concepts that processes can use to communicate about a tree. A particular process may determine that an object is a pine_tree on the basis of its temperature and the soil type around it, but it must share its information in terms of the concepts defined in the network. This approach was adopted for important pragmatic reasons — it is impossible to "define" a concept like a tree; yet information about a tree must be communicated in terms that other processes understand.

5 Other Directories

The system architecture we have described is independent of the indexing structures that are overlaid on the database; to change those structures requires only changes to the parser that processes database requests (as can be seen in Figure 1). The extensibility of the directory system allows future requirements to be accommodated without change to the overall system structure. The spatial and semantic directories were devised to allow an autonomous land vehicle to navigate through a world in which most objects are static and motion comes primarily from the movement of the vehicle itself. In other scenerios, this will be inadequate. In environments in which there are many moving objects, particularly fast-moving objects that are likely to impact the mission results, other directories that index the database through additional parameters, such as those associated with movement, are vital. The architecture described has the flexibility to accommodate such extensions.

6 Process Control

We have described the various processes that form the system as independent, asynchronous processes that can be activated be means of daemons imbedded in the database or by more conventional procedure calls. Each process uses vocabulary terms to interact with the database. Each process is continuously executing, although a process may put itself to sleep only to be awakened when predetermined data conditions exist. Who determines these conditions? Should every process be permitted to determine the conditions needed to interrupt another process? Some processes may be time critical and prefer not to be interrupted. Our approach is to require that the process itself set these conditions within the database. Any process can attach one of its daemons to any data slot of any data token, so that the process will be interrupted whenever any new or changed opinion modifies that data slot. Daemons are attached to data slots rather than data tokens because data tokens usually represent a complex item and any one process is probably interested in only some aspects of it; for example, the navigational module of an autonomous vehicle will want to be interrupted if a sensor process gives a new opinion on the position of an obstacle, but it is unlikely to need to be interrupted if the obstacle's color changes. It is, therefore, the responsibility of each process to determine when it is to be interrupted.

In a like manner, it is the process that determines what action to take when it is interrupted. The interrupt handler is part of the definition of each process. As processes are quite varied, there is no sense to the notion of a generic interrupt handler. Clearly, a process may choose to continue with what it is doing rather than to process the interrupt if it assesses the current task to be more relevent to mission success than that associated with the interrupt. Conversely, a process may instead suspend or abandon what it is doing in favor of the interrupt. The overall system concept is that of a loosely coupled system in which all processes work on their goals cognizant of the overall mission of the system. Each process determines how it can best support the mission goals and is responsible for the means to achieve this.

The process architecture parallels that of blackboard systems that were brought to prominence in the building of speech understanding systems [2]. In these systems data are placed on a blackboard; if the combination of data on the blackboard meets the preconditions for a particular procedure to execute, then that procedure is triggered and put on the schedule for computing resources. In an important way, the approach of activating processes using daemons differs from the triggering mechanism used on blackboard systems. There is no pattern matcher whose job is to trigger processes when a particular pattern of data appears in the database (or on the blackboard). For efficiency reasons, the patterns that pattern matchers are to recognize must be predetermined and compiled in at system building time. In a system that is loosely coupled, in which different processes may be present during different executions of the system — in which the system must function even if some of the processes (or hardware) fail — an approach to pattern matching that decentralizes the responsibility for determining whether a process should be triggered seems more manageable. We have chosen to trigger on an opinion being changed rather than on a particular pattern in the data itself. In selecting this mechanism, the cost of the additional processing that is done by the interrupt handler in each process was weighed against the computational cost of running a generalized pattern matcher.

In any system with multiple processes, priority will sometimes need to be given to processes that perform time-critical tasks. At other times, the system could be underutilized. As a result some processes should be scheduled as foreground jobs, which compete for resources when they request them, while others should be background processes using only spare resources. Some of the background processes have already been identified: the module that resolves data inconsistencies, the one that recovers storage space, and parts of the resource allocator itself. The system should never be idle. Computational resources are allocated to modules by a separate process, a metalevel process, that changes the time slice allocated to vari. us processes. A process that produces data, including opinions, that are used by other processes warrants more resources than a producer of unused data. In addition, a process can request more resources if it determines such a need, so that critical processes can ask for priority.

In the current system implementation all the various processes execute on one computer system, a Symbolics 3600, and all interact through a common virtual address space. This approach was adopted to eliminate the system building necessary to run experiments on multiple processors. However, the conceptual design assumes a virtual environment in which there are many processors running in parallel, with a communications network between them. This accounts for the design decision of the rather loose coupling between processes. On a network of parallel processors, we would expect some processors to be dedicated to particular modules whose computational task is matched to the pa-ticular machine hardware. Other processes would be allocated among the available processors. Although we are aware of the bottleneck that might be caused by centralizing the database, we envisage a system in which the process accepting request for database transactions will be centralized but the database itself and the procedures that carry out the internal processing may be split across processors.

7 Summary

The natural, outdoor environment in which an autonomous land vehicle operates imposes substantial obstacles to the design and successful integration of the vehicle's various sensory, planning, navigational, and control activities. The complexity of the domain and the requirement for high reliability rule out approaches that do not make substantial use of stored knowledge about the environment. An intelligent database that competently contributes to the processes that perform these various system activities is central to the overall design of an autonomous system.

The architecture of the described system is one of a collection of task experts, each implemented as an asynchronous process, that independently update the database with conclusions and deductions about aspects of the world that lie within their domain of competence. Each process must be able to take advantage of relevant knowledge that is available in the database – the knowledge system includes a means for effectively communicating information to the multiple and varied processes that wish to use it. This communication is based on a vocabulary of terms and a set of connections among them that specify the semantics of shared concepts. To support real-time operations retrieval of data from the database must be supported by database indices that match the semantics of those retrieval requests. The described database is accessed through spatial and semantic directories that organizes the various data representations to achieve flexible and timely information retrieval.

References

[1] Barr, Avron and Edward A. Feigenbaum, *The Handbook of Artificial Intelligence,* William Kaufmann, Inc., Los Altos, California, 1981.

[2] Erman, Lee D., Frederick Hayes-Roth, Victor R. Lesser, and D. Raj Reddy, The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty, *Computing Surveys,* Vol. 12, pp 213-253, June 1980.

[3] Samet, Hanan, The Quadtree and Related Hierarchical Data Structures, *Computing Surveys,* Vol. 16, pp 187-260, June 1984.

[4] Srihari, Sargur N., Representation of Three-Dimensional Digital Images, *Computing Surveys,* Vol. 13, pp 399-424, Dec 1981.

Tools and Experiments in the Knowledge-Directed Interpretation of Road Scenes

Bruce A. Draper, Robert T. Collins, John Brolio
Joey Griffith, Allen R. Hanson, Edward M. Riseman *

Computer and Information Science Department
University of Massachusetts
Amherst MA 01003

The University of Massachusetts *Schema System* is a high-level image understanding system for the interpretation of complex natural scenes. It is our position that the constraints available from general knowledge about the world can be used to organize raw image descriptions into abstract interpretations. The output of low-level image operations does not fully satisfy the requirements of natural-scene interpretation. Intermediate-level grouping procedures represent a partial solution, but are too expensive to apply indiscriminately. High-level processes use object-specific knowledge to guide interpretation by focusing attention on promising areas of the image. This paper describes the Schema System – a flexible, high-level system for the interaction of object-specific interpretation processes – and presents the results it has produced on road scenes as a justification of the knowledge engineering approach to image understanding.

1 Introduction

The University of Massachusetts *Schema System* is a high-level, knowledge-directed image understanding system for the interpretation of complex natural scenes. It is our position that the constraints available from general knowledge about objects and the structure of typical scenes can be used to organize raw image descriptions into abstract interpretations. Low-level image operations such as straight line extraction and region segmentation are meant to be general enough to serve a broad range of intermediate and high-level requirements. Consequently, their output is relatively unorganized and does not necessarily reflect the requirements of natural-scene interpretation. Intermediate-level grouping procedures organize low-level data in ways more suitable for interpretation. This is a partial solution, but many of these processes are too expensive to apply indiscriminately, and are not relevant in all situations. High-level processes use real-world knowledge to hypothesize likely objects and manage levels of belief about these hypotheses. They guide the interpretation by focusing attention on promising areas of the image and allocating resources to context-specific strategies as they become appropriate. As the interpretation proceeds, the high-level components provide feedback to low-level processes to tailor their output to the type of image and to the needs of the interpretation (see Figure 1).

With this scenario in mind, we have developed the Schema System as a flexible, high-level system to support and control the interaction of object-specific interpretation processes. This system is currently operating in conjunction with the UMass Low Level Vision System (LLVS) to identify and extract the significant objects in natural scenes. This paper describes the theory and practice of the Schema System; its architecture, its database organization and interpretation techniques, and the results it has produced in the domain of New England road scenes. We present the experimental results as a justification of the approach, and to demonstrate that the required knowledge engineering task is feasible.

1.1 Overview

The task of high-level vision is to identify meaningful objects and relations in natural scenes, and ultimately to generate a three-dimensional interpretation. Meaningful objects include those which a person might identify from the image, as well as objects which must be identified for a particular task such as navigation. Meaningful relations are those spatial relationships which serve to locate objects relative to each other, and to locate the scene relative to the observer. A high level vision system should identify the semantic objects in the scene and gather information about them.

One key to the construction of an image understanding system is the use of general knowledge about the world to guide the interpretation of image events. Low-level processing on a typical natural scene produces thousands of lines, hundreds of regions, and an arbitrary number of other detectable events. Not all of these *tokens* are significant. Intermediate-level grouping processes partially alleviate this problem by combining the low-level data into a smaller number of relatively salient events. However, these processes are typically too expensive to apply everywhere. Moreover, they are subject to the same problems that afflict the low-level processes – the relevance and proper organization of the events they detect are dependent on the context in which they occur. High-level control is needed to focus such processes on a limited number of events under the appropriate circumstances.

In addition to focusing the attention of lower level processes, a high-level vision system must evaluate the extracted events in terms of its interpretive context. For example, merging adjacent regions depends on more than their measurable attributes in feature space. The decision should also be based on the image characteristics of the object involved – high contrast lines are commonly found on the texture boundaries of foliage regions, and thus should not inhibit region merging if the entire tree crown is desired, whereas the presence of high contrast lines along the edge of a tree trunk region generally signals the boundary of the trunk. Object-based hypotheses also provide access into a library of object-specific interpretation strategies.

*This research was supported by DARPA under contracts N00014-82-K-0464, DACA76-85-C-0008; and by AFOSR under contract F49620-86-C-0041.

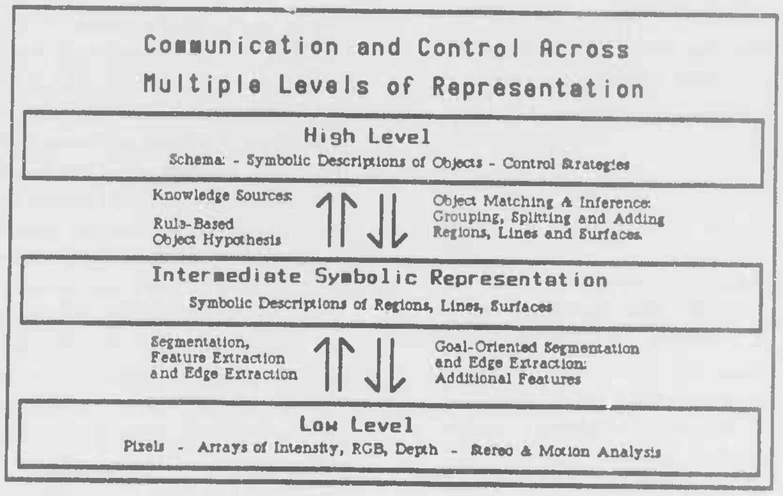

Communication and Control Across
Multiple Levels of Representation
High Level
Schema: - Symbolic Descriptions of Objects - Control Strategies
Knowledge Sources:
Rule-Based Object Hypothesis
Object Matching & Inference: Grouping, Splitting and Adding Regions, Lines and Surfaces.
Intermediate Symbolic Representation
Symbolic Descriptions of Regions, Lines, Surfaces
Segmentation, Feature Extraction and Edge Extraction
Goal-Oriented Segmentation and Edge Extraction: Additional Features
Low Level
Pixels - Arrays of Intensity, RGB, Depth - Stereo & Motion Analysis

Figure 1. Multiple levels of communication.

1.2 The System

The Schema System applies real-world knowledge to the interpretation of natural scenes. The knowledge base is composed of *schemas* and *knowledge sources*. A *schema* is a template for recognizing some class of objects ("object" is used in a general sense that includes road scene and outdoor scene). A schema can be thought of as a frame, packaged with its own interpreter. When the current context predicts that a given object instance might be present, a new schema instance is *instantiated*. A schema instantiation is a copy of the schema template which runs as an independent agent with its own process and local state. The instance processes image data concurrently with previously activated schema instances, and can communicate with them through a central blackboard. A schema instance has two primary tasks, *control* and *synthesis*. Control involves the sequence of actions that a schema instance will take, such as determining which knowledge sources to run, and when and where to run them. Control also involves dynamic resource allocation, which we have not yet experimented with. Synthesis is the process of integrating knowledge source results into a single hypothesis, and linking individual hypotheses into a consistent interpretation network.

Knowledge sources are image interpretation utilities available for invocation by schema instances. They are generic tools for performing common actions such as graph matching and feature comparison. They have a simple parameterized control structure, and they maintain no internal state between invocations. Two knowledge sources are discussed later in this paper: Rulesys, a feature-based initial hypothesis generator, and OVARC, a system for *O*bject *V*erification by the *A*pplication of *R*elational *C*onstraints.

1.3 The Application of Knowledge

During an interpretation, schema instances will employ many different types of knowledge. The types of knowledge currently used include: 1) two-dimensional and three-dimensional models (telephone pole, road sign), 2) feature space characteristics (initial hypotheses), 3) part-subpart and ISA relations (road sign/pole, road/roadline), 4) co-occurrence heuristics (gravel next to road), and 5) relative size, position, and location (sequences of telephone poles lining the road). The level of description that this knowledge is concerned with is much higher than the level of abstraction at which low-level processes operate. Central to the system, therefore, is a database called the Intermediate Symbolic Representation (ISR). The ISR represents image data in terms of symbolic *tokens*. The simplest tokens are output by the low-level line extraction and segmentation algorithms (other forms of low-level data, representing depth, motion, etc., can also be stored). Others are "hallucinated" by high-level processes. The most sophisticated tokens are typically defined in terms of other tokens, by iterative processes that build more and more abstract tokens from the database. The rectilinear line grouping system of Reynolds and Beveridge is a good example of this [Reyn87].

The interpretation process proceeds opportunistically. The system has certain initial expectations, such as being outdoors and on a road, which are realized through starting the system with these schema instances already active. These then predict the presence of objects by starting schemas to recognize them. Those that fail to generate initial hypotheses terminate, the rest attempt to verify their hypotheses, possibly predicting new objects and starting new schema instances along the way. This leads to a combined bottom-up and top-down interpretive approach, in which those characteristics of objects that are cheap to compute are used to generate initial hypotheses, and the attempt to verify or negate these hypotheses then controls further processing. Schema instances whose objects are complimentary communicate this to each other as a source of support. Those whose interpretations contradict each other compete. The final interpretation emerges as a network of consistent and believed object hypotheses.

The remainder of this paper presents the major components of the system in more detail, and describes a set of experiments using this system to interpret New England road scenes. The Schema Shell, the ISR, and the Rulesys and Object Verification (OVARC) knowledge sources are each presented in greater detail. Then the results are presented, including details of some of the schemas and knowledge sources used, and traces of intermediate results showing how the final interpretation was assembled.

2 Tools

2.1 The Schema Shell

The *Schema Shell* is a tool that supports the construction of large sets of schemas [Drap86]. The road scene interpretation schemas presented in this paper form one such set; a set of schemas for house scenes was previously developed. The shell provides a simulated distributed environment (until parallel hardware arrives) in which any number of schema instances may run concurrently. Communication between instances is implemented through a central *blackboard*. While each schema instance executes in its own memory space, all instances may write to and read from the blackboard. This provides a uniform, asynchronous communication mechanism between schema instances.

The notion of schemas supported by the Schema Shell is the result of a long line of research at the University of Massachusetts [Arbi78], [Arbi81],[Hans78],[Weym86]. The earliest work was biologically motivated and theoretical in nature [Arbi78], [Arbi81]. The first implementations of schemas, and the first use of schemas in vision, were similar to a frame based system [Hans78]. More modern implementations have focused on the aspects of concurrency and the active nature of the agents [Weym86]. Schemas as defined by the Schema Shell are both a simplification and a generalization of the concept used by Weymouth. Major differences include the adoption of a single, general purpose communication mechanism, the separation of local memory from communication, the formalization of the notion of strategy, and a simplified method of schema creation.

A schema requires local memory and a process to operate on it. The former is specified by the user as a list of schema variables; these may either have initial, object-specific values (such as a two-dimensional model for the OVARC knowledge source) or may be left to be filled in with image-specific data. The latter is provided by the user in the form of an *interpretation strategy*. This is a Lisp procedure which invokes knowledge sources in response to the current context, and which stores the resulting data in the schema variables. When it is possible to perform aspects of the recognition task in parallel, the strategy may spawn other strategies which operate concurrently on the same local memory. This provides another level of parallelism below the schema level. In the experiments described below, each schema was in fact implemented as up to seven independent strategies. When a schema instance predicts the presence of a related object in the image, it invokes a new schema instance which is allotted a distinct section of memory. The new instance pursues its hypothesis independently of its parent. In this way, the system will have many concurrently executing schemas, and within each schema, several concurrent strategies.

Schema instances communicate through a central blackboard. At any point during its processing a schema strategy may write an arbitrary message to the blackboard. Any other schema is then free to read, erase or modify that message. This provides a single, uniform communication mechanism which can be easily implemented on a variety of distributed architectures. To post a message, a schema instance needs only a section name to write to. The mechanisms for reading, erasing and modifying messages are of necessity more complex. To read a message, a schema instance must specify the section to be read from, and some method for selecting among the messages present on that section. The selection criterion is provided in the form of a predicate; all messages for which the predicate evaluates *true* are returned by the read function. Time of posting may also be used as a selection criterion. If no messages are found, there are two possible actions. The read function can return null, or it can suspend the calling strategy until a suitable message is posted. Since both behaviors are useful, the Schema Shell provides two basic reading functions, *read-or-nil* and *read-or-wait*. If no messages are found, the former returns null, while the latter suspends the caller. A suspended instance is said to be *sleeping*, and consumes virtually no resources until a suitable message is written and the instance is *awakened*. The shell provides routines for removing messages from the blackboard as they are read (*erase-or-nil* and *erase-or-wait*) and for altering existing messages (*modify-blackboard*).

The Schema Shell additionally provides I/O routines and debugging aids. These include tools for identifying the status of schema strategies, inspecting schema instances' local memory, and displaying ISR tokens to either black-and-white or color monitors. Using a mouse-driven display during debugging helps the programmer to visually locate and identify a desired token from among hundreds.

2.2 The Intermediate Symbolic Representation

Input to the Schema System consists of abstract image descriptions which are produced from low-level processes of line and region extraction, and from intermediate-level processes of grouping and selection. These image descriptions are stored in the Intermediate Symbolic Representation. The ISR is a database which has been custom-built for the efficient storage, manipulation, and retrieval of abstract image data. The fundamental unit of representation is the *token*. Each token has a unique name and a list of feature slots. The ISR can be used to store anything that can be characterized by a list of features and values; some of the image events currently stored include region segments, extracted edge lines, fields of homogeneous texture, rectilinear line groups, and region-line relations. The benefits which result from imposing a uniform representation and user interface on all intermediate level tokens are enormous. It is now natural to think in terms of multistage and hierarchical grouping processes which take in tokens at one level of abstraction and produce tokens at the next higher level [Reyn87], [Weis86]. The freedom from having to represent image events by sets of pixels makes it easier to store relational tokens, that is, tokens which represent the relationships between other tokens [Belk85]. The ISR also facilitates the sharing of results between researchers, and between systems — it is implemented in both the LLVS and the old VISIONS system, and results produced by one can be read in to the other thanks to the ISR's standardized data format.

An organized environment is maintained in the ISR by partitioning the total set of tokens. At the highest level, tokens are partitioned according to the *image* they were extracted from. There is also an intermediate level of partitioning called the *tokenset*. All tokens in a tokenset must have the same features defined for them, so in a sense they are all of the same "type." For instance, a standard line tokenset contains features which have meaning when applied to lines, like length, orientation, and contrast. Tokens in a region tokenset have different, region-specific features, like average intensity, compactness, and euler number. By convention, tokensets contain all the tokens produced by the same abstraction process running on a single image, but tokensets can also be used to impose arbitrary groupings of tokens of the same type, such as dividing them up by spatial location.

Each feature associated with the tokens in a tokenset has a name, a value slot, a data type, and a computation function. Supported data types include numeric fixed and floating point, as well as a pointer data type for representing strings or lists of other tokens. Since many tokens have a physical realization in an image, a special bitplane data type is provided for representing the subset of image pixels which are associated with a token. Operators exist for taking the intersection, union, and difference of token bitplanes; bitplanes can also be created from scratch by specifying a list of bounding vertices. This directly supports strategies for fusing information across multiple representations. For example, relations between lines and regions can be derived by creating line bitplanes and intersecting them with region bitplanes and analyzing their overlap [Belk85].

Two of the more interesting aspects of the ISR involve the way features are accessed; these methods include associative access and on-demand computation. Associative access complements the normal data access mechanism (which can be loosely translated as "give me the length of line 45 in image I") by allowing the ability to access tokens by attribute ("give me the lines in image I whose length is between 5 and 10"). The present associative capabilities include coarse spatial location ("give me the lines in image I whose bounding rectangle overlaps X", where X is a specification of some rectangular window in the image), conjunctive filtering on ranges of numeric feature values ("give me the long, high contrast lines"), and limited disjunctive filtering ("give me the long, high contrast lines with an orientation of close to 90 or close to 180 degrees" – that is, vertical and horizontal, long, high contrast lines).

The other interesting access mechanism in the ISR is on-demand feature computation. When a feature with an undefined value is accessed, the computation function associated with that feature is invoked. The function may choose to store the value, along with any intermediate or related values that have been computed, in which case the value is retrieved directly the next time the feature is accessed. If the value does not get stored, that function will be invoked again when the feature is accessed again. Whether or not a value was directly retrieved or had to be computed is transparent to the user (other than in terms of visible speed of the process). This facility for on-demand feature calculation has three important benefits. One, only the features that are actually going to be used need to be calculated; two, it is possible for a schema instance to define new features or redefine old ones dynamically during the course of an interpretation; and three, new tokens can be introduced into the database at runtime, and they will be essentially indistinguishable from pre-existing tokens.

This last point bears repeating. The ISR allows schema instances to create tokens during an interpretation, create their bitplanes either from scratch or as some combination of token bitplanes, and access their features, at which time the new feature values will be calculated automatically. It is thus possible for the schema system to dynamically "correct" misleading segmentations based on combinations of top-down knowledge, and to "hallucinate" regions based on structures found in the line data.

2.3 The Rulesys Knowledge Source

A high level vision system requires at least one component knowledge source for comparing measurable attributes of image events to expected attributes of known objects in the world. At present, the Griffith Rule Generation System, or Rulesys knowledge source, fills that role for the Schema System. The Rulesys knowledge source constructs rules which map feature values into object likelihood scores. The resulting scores suggest initial hypotheses which the Schema System considers when evaluating promising paths of inquiry.

The Rulesys applies an automatic rule construction method similar to [Lehr87] and [Belk85]. The input to the system is a list of significant objects, a set of training images in which those objects have been hand-labelled, and a list of the measurable attributes over which the rules will be defined. The resulting rules distinguish one object from another in the knowledge base. The Rulesys knowledge source operates in an environment of unreliable feature measurements and imperfect *a priori* information about object attributes. To ensure robustness, many partially redundant features combined to provide more reliable results. A simple rule maps single feature values into object likelihood scores. Complex rules group simple rules into general classes such as color and texture. Scores are combined to produce a single score which relates an object class to a feature category. Currently implemented feature categories are color, texture, shape, size, and location. For the New England road scene experiments presented in this paper, only color and texture rules were used.

2.4 The OVARC Knowledge Source

Once the Schema system has established a set of initial hypotheses, it is necessary to invoke verification strategies to confirm or reject the labelling proposed by each hypothesis. These strategies in turn rely on knowledge sources (KSs) which are general-purpose modules whose output will affect the system's confidence in a hypothesis. One of the knowledge sources we have been experimenting with performs object verification by the application of relational constraints (OVARC). The purpose of OVARC is to take a structure or template called an object description (OD) and to find a set of tokens which match the template, attribute for attribute, relation for relation. The attributes (1-place predicates) and relations together constitute the constraints on the tokens which define an instance of the OD in the image.

An OVARC object description is a graph. Each node of the OD represents a token. Attribute constraints are represented as an arc from the node to itself. Relational constraints are represented as a directed arc from the node to another node in the graph. The name of an arc is either the name of a feature stored in the database (ISR) or of a function which computes the constraints as each candidate data set is processed. This gives the user a great deal of flexibility in the choice of constraints for the object description. The description may contain any relation or constraint which is represented in the database or which can be computed from the database or from the interpretation context. The object description structure controls the allocation of processing resources by defining the order of application of constraints.

The output of OVARC is either a signal of failure to match or an new aggregate token which represents the object defined by the OD. This new token will have attributes and relations of its own, one of which is the internal structure which qualifies it as a match to the OD.

The input tokens for an OD match may be of different types (regions, lines, groups, other objects) or they may all be of the same type. It is possible to have a description of an outline view of a telephone pole or road sign (lines) or a description which includes a piece of road (region), pieces of road edge (lines) and pieces of a centerline (lines and region).

The final step in the matching process is a subgraph isomorphism search [Ulln76]. Taking a set of data tokens with their attributes and relations as the candidate data graph, the goal is to find one (or more) subgraphs which are isomorphic to the object description graph. All solutions to the subgraph isomorphism problem exhibit exponential behavior. Essentially we have taken a method which is exponential in n, but which is otherwise adequate in expressiveness and generality, and restricted its use to situations in which n is very small. In the worst case the complexity of the subgraph isomorphism algorithm is $\binom{m}{n}^n$ where n is the number of nodes in the object description graph and m is the size of the candidate data graph. For this algorithm to perform reasonably the number of nodes in the OD should be very small (less than 10, preferably less than 6) and the constraints in the OD should reduce the candidate data graph to a relatively small size as well ($\sim n^2$). The size restriction on the OD graph is not a serious restriction since a complex object description can be decomposed into a hierarchical set of ODs.

Given an OD of the proper size, it is necessary to reduce the size of the candidate graph. Since OVARC is an object verification KS, it will be invoked only when there is a reasonable likelihood of finding the object, so the search space will already have been reduced from the whole image to a few locations in the image. Furthermore, in creating an OD, a Schema designer has a great deal of flexibility in controlling computation. To begin with, there are several preliminary filters which can be invoked to reduce the search space. Before the matcher is run, every effort is made to reduce the size of the candidate graph. For each node in the OD, constraints are specified which reduce (1) the number of tokens which can match this node (via attribute constraints) and (2) the number of tokens which are candidates for related nodes (via relational constraints). The nodes are processed strictly according to the order specified in the OD, so that knowledge of the relative strength and computational cost of individual constraints can be reflected in the order of their application. Nodes with strong constraints are listed first in the description. Nodes with constraints which require more computational resources are listed later when presumably fewer candidate tokens will have survived to be processed. Due to the progressive filtering of the candidates, constraints will be computed only for those candidate tokens which have some chance of matching nodes in the OD. Proper timing of these calculations gives a researcher the ability to fine-tune the description to get the best matching behavior for the smallest amount of computation.

The calculation of some constraints may be so costly that an OD will defer their computation until matches have been found. A post-filtering facility is included for this purposes.

Since the heart of this object verification KS is an algorithm with exponential behavior, after all elimination of candidate nodes is complete and the matching process is about to begin, if the size of the candidate set exceeds a user-specified limit, OVARC reports a *computational complexity failure*. This failure can be interpreted in one of two ways: (1) either the object description is too weak, i.e., it admits too many potential matches, or (2) this knowledge source is not able to handle this particular set of data. The Schema system should be able to use this information to dynamically adapt its interpretation methods. Failures of type (2) are closely related to "normal" failure, i.e., failure to match the object description. A successful match is very strong support for the presence of a given object in the image, but a failure to match may mean the object is absent or that it is present but occluded or viewed at an unusual angle. The only convincing evidence that an image event does not represent some particular object is a verification that it in fact represents a different object.

We are currently expanding the set of descriptions and the capabilities of the description language. One feature which appears to be useful is the ability to specify "non-unique" nodes in the object description. For example, one of the line-extraction operators that we use tends to fragment lines such as the crossbar edges in a telephone pole (see example OD). A description could specify fragmented collinear lines with a "could-also-be-node-i" relation, so that two nodes could each represent a token (i.e., there are two line fragments) or both nodes could be matched by the same token. In general, however, matching partial or incomplete views is part of the larger high-level vision problem. The OVARC knowledge source provides some flexibility in describing objects, but further research in grouping will have to uncover ways of conveying information from good partial matches to higher-level processes.

3 Experimental Results

The aim of this experiment was to demonstrate the feasibility of the knowledge based approach to vision by showing that 1) constructing a knowledge base for a domain is a manageable task, and 2) the use of this knowledge produces significantly improved image interpretations. Section 3.1 discusses the knowledge base developed for interpreting road scenes. It is the product of a small ($2\frac{1}{2}$ man) development team working for approximately one month. The team had available to them knowledge sources from the house scene domain; the assembly of a library of useful knowledge sources from many different domains is an ongoing aspect of this work. Section 3.2 describes how the schemas outlined in 3.1 produced the interpretation results shown below.

There are a total of six road scene images. In all six, the camera was level to gravity. In five of the six, it was positioned as a vehicle in the road. Figure 2 shows a sample scene. The poorly defined road edges, heavy shadowing and extreme depth of field are all typical of the domain. Interpretation results for this image are presented in Figures 3a and 3b. Figure 4 is an image from the set which has an uneven ground plane. This complicates 2D interpretation, and will present a challenge to

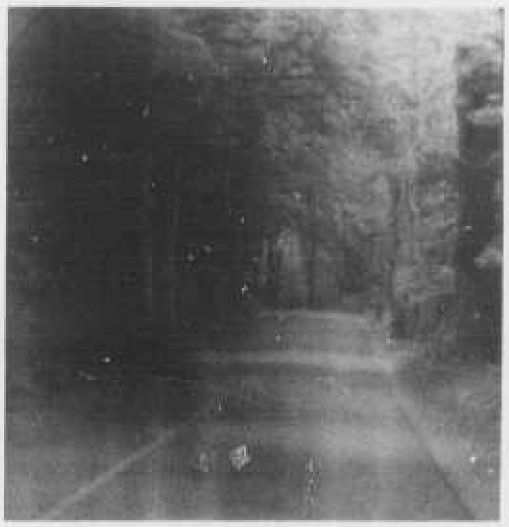
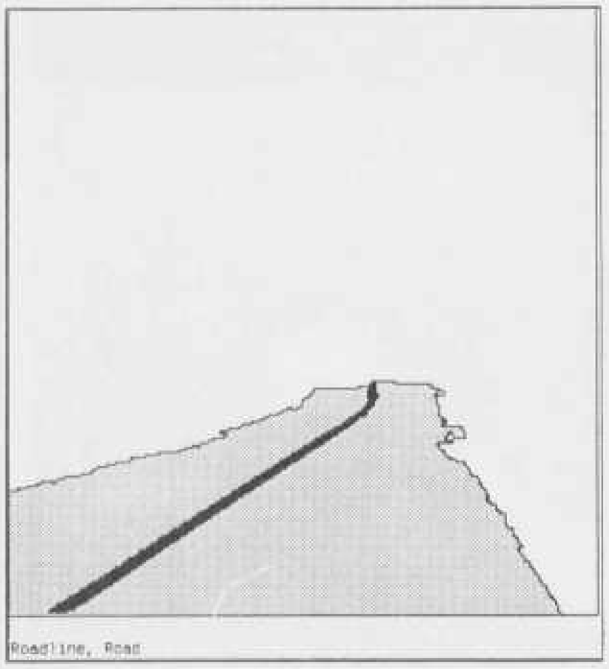


Figure 2. Sample road scene photograph.



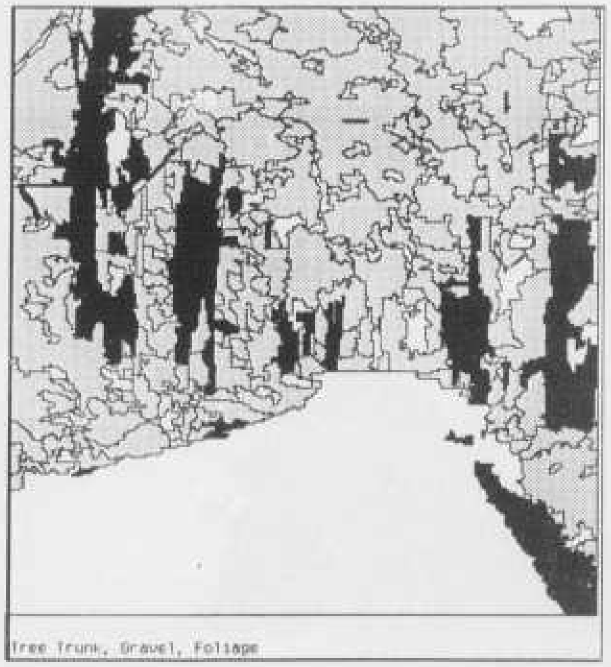


Figure 3a,b. Interpretation results for Figure 2. (a) Road line: black; road: light grey. (b) Tree trunk: black; gravel: dark grey; foliage: light grey.

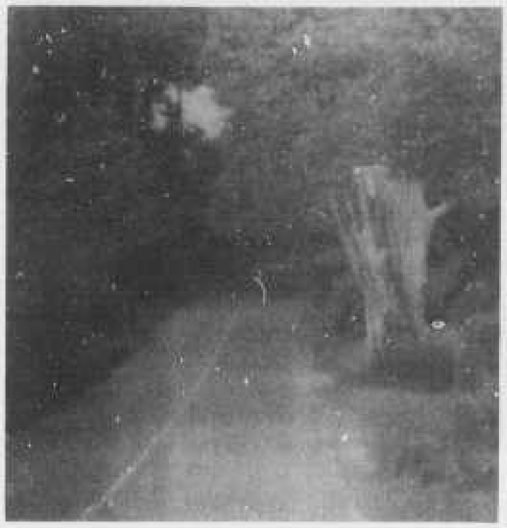
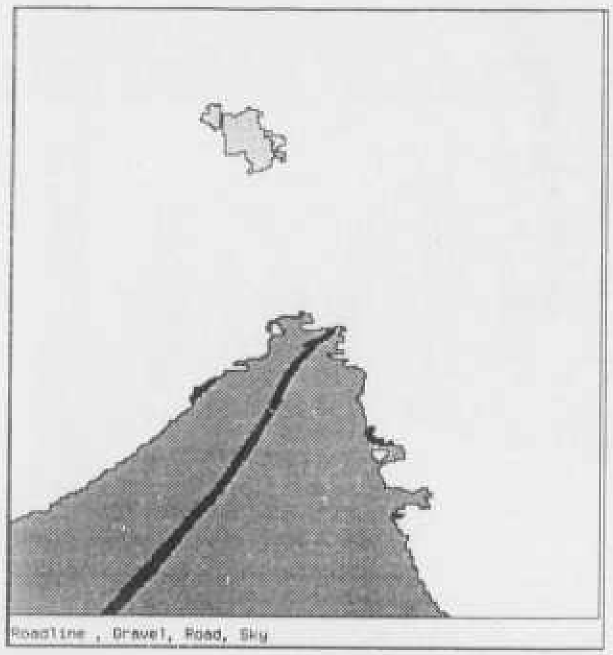


Figure 4. Road scene with an uneven ground plane.



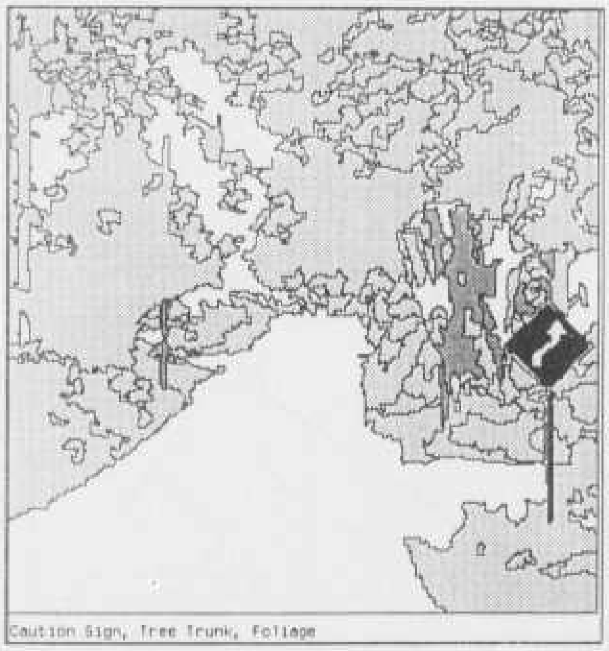


Figure 5a,b. Interpretation results for Figure 4. (a) Road line: black; gravel: dark grey; road: medium grey; sky: light grey. (b) Caution sign: black; tree trunk: medium grey; foliage: light grey.

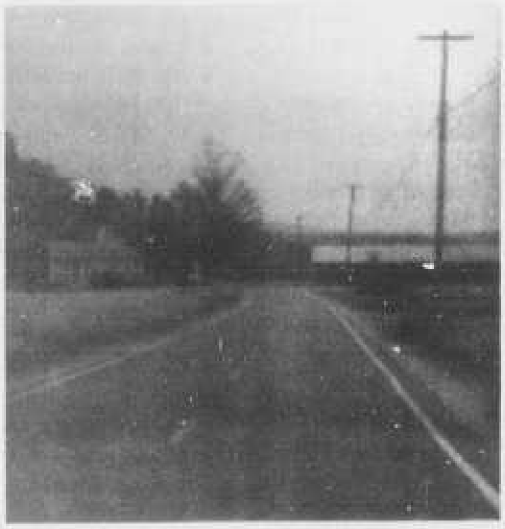


Figure 6. Road scene with other man-made structures.

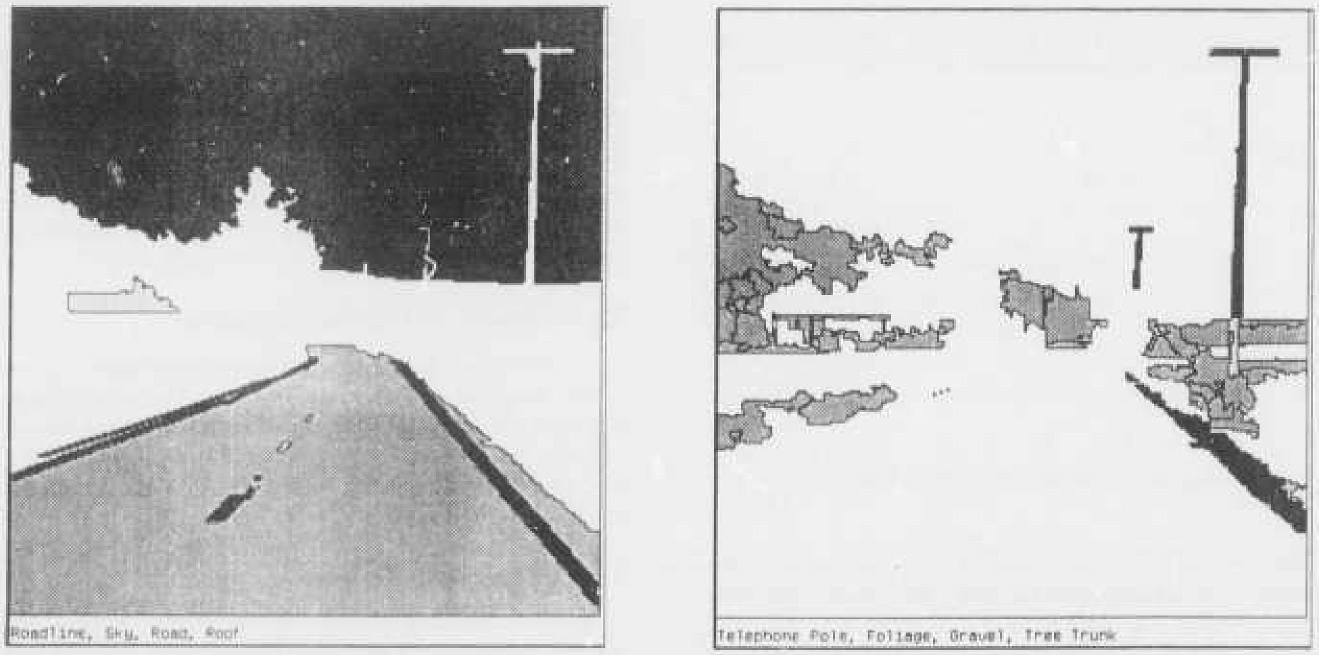

Roadline, Sky, Road, Roof
Telephone Pole, Foliage, Gravel, Tree Trunk

Figure 7a,b. Interpretation results for Figure 6. (a) Road line: black; sky: dark grey; road: medium grey; roof: light grey. (b) telephone pole: black; gravel: (dark grey): foliage: medium grey; tree trunk: light grey.

future 3D reasoning extensions. The system's interpretation of this image is presented in Figures 5a and 5b. Finally, Figure 6 shows an image with more man-made structures. Figures 7a and 7b report the schemas' conclusions about this image.

There are two interpretation errors due to incomplete schema knowledge. In Figure 7a, the detection of the roof supplies enough information to identify the house wall below, but there is no building schema to accomplish this.

In Figure 7b, the bottom of the telephone pole is incorrectly labelled tree trunk. This error will be corrected by an object-completion knowledge source that extends the telephone pole model after OVARC has found it. This is easily done by following the linear structure of the pole.

3.1 The Knowledge Base

There are currently 12 classes of objects in the knowledge base: sky, foliage, tree trunk, road, roadline, roadside gravel, telephone pole, telephone wire, roof, stop sign, yellow (caution) sign, and road scene. A decision was made not to include field or grass, since only one instance of each appears in the image set. The objects are organized in a *calling hierarchy* (Figure 8) which indicates which schemas may invoke what other schemas. This generally follows the traditional part/subpart hierarchy, but it is not compelled to do so. Any information about one object that can be used to predict another should generate a connection in the calling hierarchy between the two objects. In this experiment, the calling structure was strictly hierarchical; this avoided problems with generating redundant schema instances. Our impression is that future knowledge bases will require a tangled calling hierarchy.

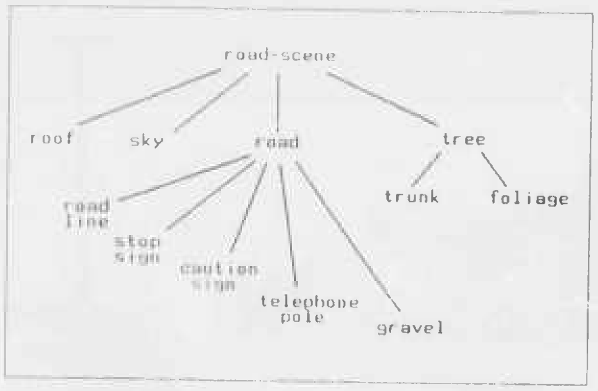


Figure 8. Schema calling hierarchy.

How to combine evidence from different knowledge sources and propagate evidence between objects is a difficult, unanswered problem. In this experiment every schema maintained both an internal and an external hypothesis. The internal hypothesis is kept in the schema's local memory, and is used for recording object specific symbolic endorsements. The external hypothesis resides on the blackboard, and has a confidence value on a simple, five-valued scale, *'no-evidence, 'slim-evidence, 'partially-supported, 'belief* or *'strong-belief*. The endorsements of the internal hypothesis are symbolic results returned by knowledge sources. The road schema, for example, recognizes the symbols road-color, road-texture, centerline-found, left-line-found, right-line-found, gravel-found, and side-structure-found (the last refers to caution signs, stop signs and telephone poles). The schema uses an object specific routine to map the endorsements of the internal hypothesis onto the confidence values of the external hypothesis. This preserves modularity by allowing one schema to inspect another's external hypothesis without knowing the details of its construction. While it is too early to draw any firm conclusions about this mechanism for combining evidence, its performance in this experiment has been encouraging.

The internal structure of each schema was organized according to a schema template (Figure 9). This divided the recognition process into seven subtasks, each handled by a different strategy. The OHM, or *O*bject *H*ypothesis *M*aintenance strategy, provides the interface to other instances. It creates and maintains the external object hypothesis, and reads the external hypotheses of other instances from the blackboard. The Initial Hypothesis strategy provides a focus of attention mechanism. It generates inexpensive but plausible internal hypotheses for other strategies to corroborate or deny. The other strategies are Extension, which extends the current hypothesis based on partial results; Support, which exploits redundancy by seeking additional evidence for hypotheses; Negative Information, which must reason about any internal failures or unexplained data; Conflict, which handles inconsistencies with the external hypotheses or other instances; and Subpart, which implements the calling hierarchy discussed above.

Certain strategy types were more successful than others in this experiment. No negative information strategy has yet been implemented. The extend and support strategy types were both heavily used, but appear to fulfill a similar function; no single schema uses both. It is also necessary that different schema's conflict strategy be mutually consistent; to this end, a single, generic conflict strategy was developed and used for all objects. The OHM strategies, although individualized for each schema, were all very similar. Their code could be replaced by a simpler, declarative format.

3.2 Examples of Schema Behavior

At some level, the behavior of a heuristically based program is best described on a case by case basis. This section shows examples of how different objects are recognized in the Schema System. Included are interpretations of both man-made (e.g. road, caution sign) and natural objects. We also discuss areas in which the system needs to be extended.

3.2.1 Recognizing Road

Road is recognized through the interactions of the *road* and *roadline* schemas, with *gravel*, *caution sign*, *stop sign* and *telephone pole* also available to provide support. The initial hypothesis strategy for road uses an initial hypothesis generation tool similar to that discussed in [Lehr87] known as the Rulesys. The initial hypotheses generated for three images are given in Figure 10. In general, road sections near the observer are found. However, in Figure 10c the system also puts forth the sky as a road hypothesis, and many more distant road sections are usually missed, as is the case in 10a and 10b. The regions in-

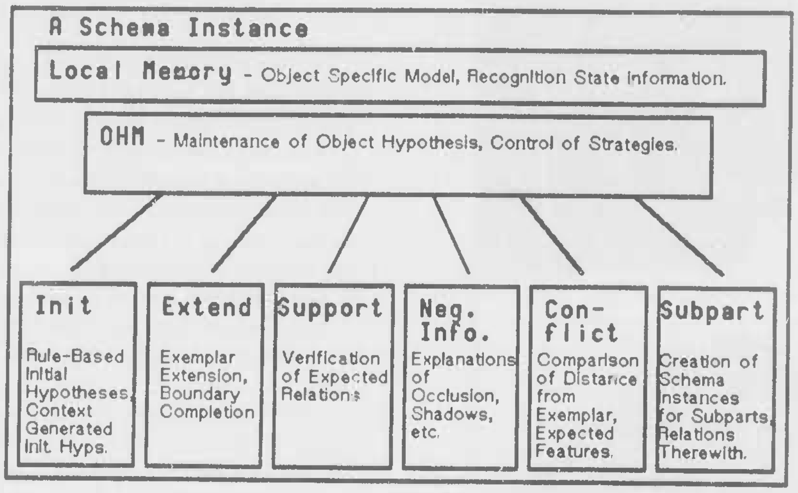
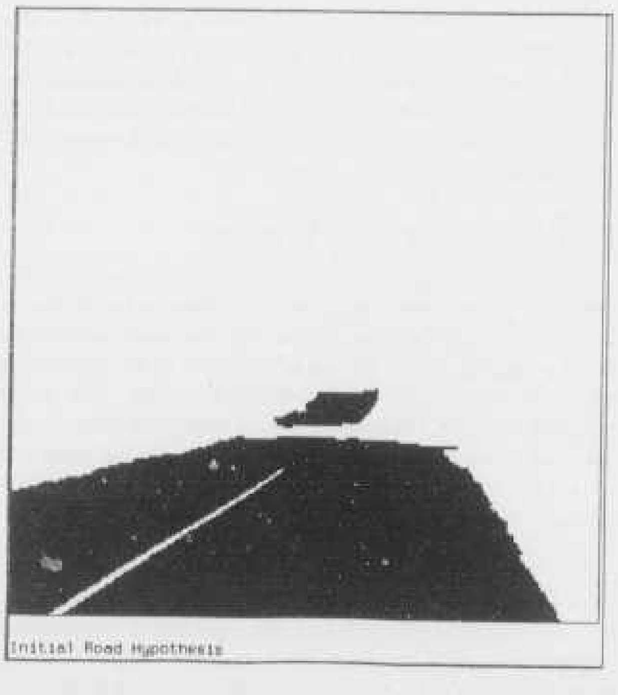

A Schema Instance
Local Memory - Object Specific Model, Recognition State Information.
OHM - Maintenance of Object Hypothesis, Control of Strategies.
Init
Rule-Based Initial Hypotheses, Context Generated Init. Hyps.
Extend
Exemplar Extension, Boundary Completion
Support
Verification of Expected Relations
Neg. Info.
Explanations of Occlusion, Shadows, etc.
Con-flict
Comparison of Distance from Exemplar, Expected Features.
Subpart
Creation of Schema Instances for Subparts, Relations Therewith.

Figure 9. A Schema template – the OHM.

dicated by the Rulesys are grouped by the initial hypothesis strategy, which causes the OHM to activate the road subpart strategy. This in turn activates roadline schemas focussed on any long, high contrast lines that overlap the road hypothesis. It also invokes a schema to look for gravel (of the type often found on roadsides) along the edges of its hypotheses.

The interplay between the road and roadline schemas is quite involved. The roadline schema (like all schemas) attempts to find support for its hypothesis. One pair of endorsements which fosters belief is 1) feature space support from the Rulesys, and 2) a bounding set of parallel lines. These two endorsements, coinciding with a partially supported road hypothesis, will lead the roadline schema instance to have confidence **belief* in its external hypothesis. It will also have **belief* in a hypothesis that has not been supported by the Rulesys if the corresponding road hypothesis is strongly believed. The



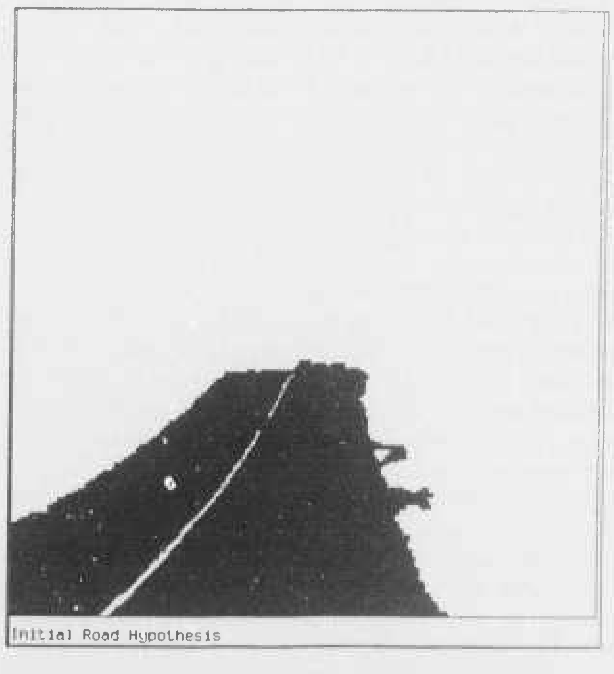


Figure 10a,b,c. Initial road hypotheses. (a) Image in figure 2.
(b) Image in figure 4. (c) Image in figure 6.

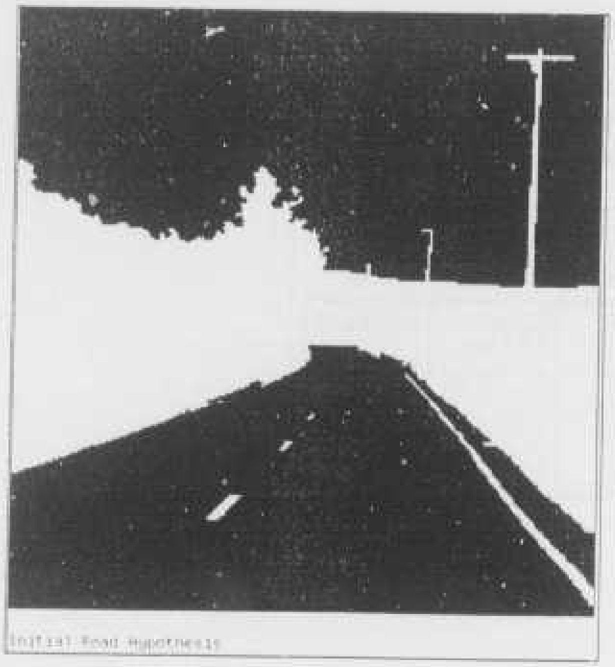


Figure 10c

item most likely to foster belief in a road hypothesis, however, is a believed roadline in the appropriate position (identified as either centerline, left sideline or right sideline). Hence finding the first segment of roadline increases the system's belief in other complementary roadline hypotheses. Figure 11 shows how these factors interact in an actual image.

Once road and roadline have been identified, their hypotheses can generally be improved. While sections of roadline near the camera tend to be demarcated by strong parallel lines, the quality of these lines deteriorates as the road recedes. Often, the line extractor will lose portions of one or both lines. In addition, projections of roads in images often bend, either because the road turns or because the actual ground plane is not flat. This inevitably leads to broken lines. However, once a section of a roadline has been identified, this knowledge can be used as an *island of certainty* to extend the roadline hypothesis past the point where the line structure breaks down. This in turn allows the hypothesis for road to be extended, since the relationship between the road and roadline is established (centerline, right sideline or left sideline). Figure 12 shows the road hypotheses before and after expansion.

The road schema also interacts with the gravel, telephone pole and sign schemas. Here, however, the interaction is weaker. In the case of gravel this is because 1) many roads in the world are not lined with gravel, and 2) the system currently lacks a method to verify an initial gravel hypothesis. As a result, gravel hypotheses are only believed once a neighboring road has been confirmed. In the case of signs and telephone poles, these objects can be reliably found, but establishing their relationship to the road hypothesis using only 2D reasoning is difficult. Here again, future work in 3D reasoning should help.

3.2.2 Recognizing Telephone Poles

In this set of experiments on road scenes, the basic assumption is that the camera is in the position of a vehicle. Consequently, schemas are looking for telephone poles along the side of the road in the foreground and middleground. The 2d model for a pole is a simple 'T' made by an upright and a crossbar. The object description for such a model contains four nodes, for the edges of the post and the crossbar. The model ignores short edges on the ends of the crossbar and the post. The code for the object description is given in Figure 13.

Once one or two poles have been identified, perspective information can be used to hypothesize the location and size of additional poles.

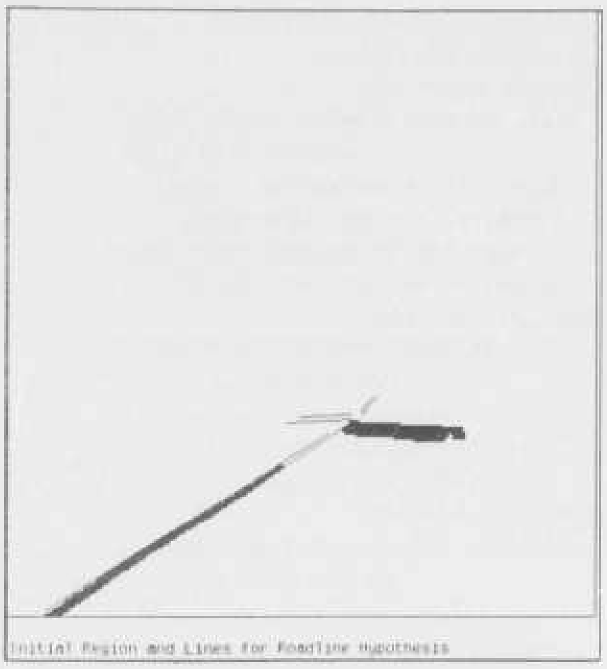
Let us now consider a sample run of an telephone pole object description. In Figure 14a, a set of candidates have been selected by location and orientation. Nineteen groups out of a possible 323 were selected by orientation alone. In Figure 14b, on-demand constraints have been computed, paring down the set of candidate lines from 266 to 10, an average of 2.5 lines per description node. This reduction is due to the vertical and length constraints on the post lines. percolating through the rest of the graph. Since verticality and size are the strongest constraints in the description, selection of candidates for the post reduces the amount of computation necessary to eliminate candidates for the crossbar edges.

The Figure 14c shows the output of the subgraph isomorphism computation. There were no false positives for this image. The foreground telephone pole is easily matched because of the strength of its lines and the regularity of its relations with respect to the object description. The success in finding the middle-ground pole is due the strength of the line-extraction process [Weis86] and the effectiveness of the grouping methodology [Reyn87].

Once the object has been found, the model can be completed by the telephone pole schema instance. Knowledge about size, symmetry and positional relations can be used to fill in the edge information, confirm region identification and contribute support to hypotheses about road location and direction, and ground plane irregularities. (Figure 5b)

3.2.3 Recognizing Road Signs

Road signs are in general easily found by their bright colors. These initial color-based hypotheses can then be verified by locating the supporting post. Since the hypothesis determines the location and size of the post, finding it is an inexpensive task. The Schema System misses one stop sign. The image in Figure 15 contains two distinct stop signs; one that is large and close to the viewer, and another that is farther away and at an oblique angle. The nearby sign is easily and correctly found. The other is proposed as an initial hypothesis, but cannot be confirmed because the linear structure of the post is missing. This is a case where knowledge directed control of the low level processes is called for. The system knows where to focus its attention; but although the capability for extracting the post through a top-down line extraction routine is being developed, it has not yet been integrated into the Schema System [Kohl87a][Kohl87b]. Figure 16 shows the initial stop sign hypotheses and the single, final believed stop sign hypothesis.



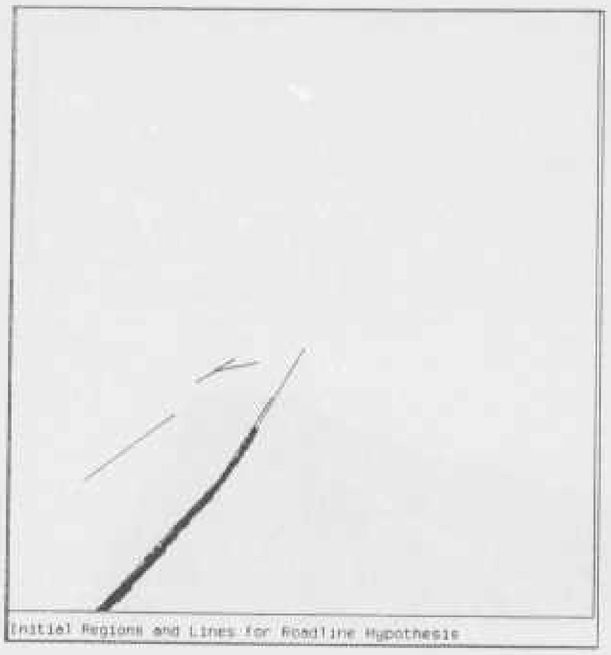
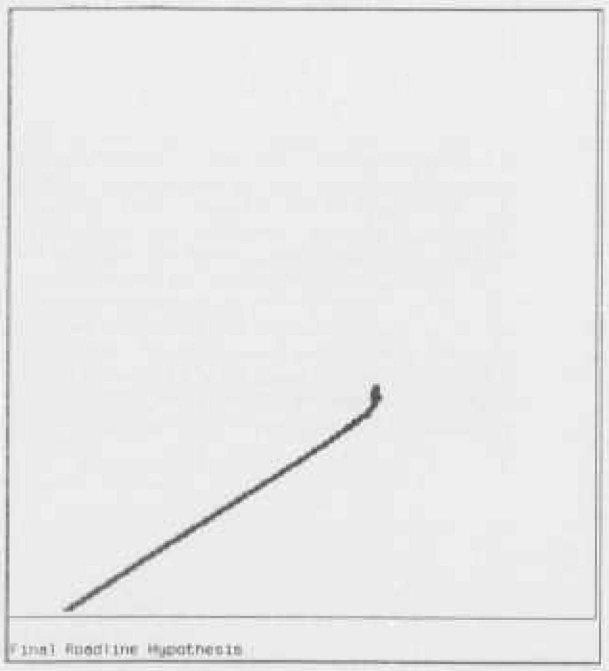


Figure 11a,b. Interaction of lines and regions in initial roadline hypotheses.



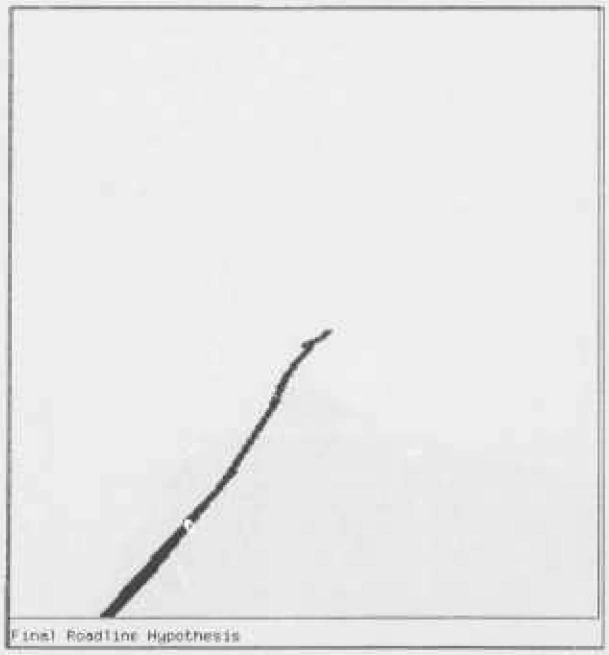


Figure 11c,d. Results of roadline expansion.

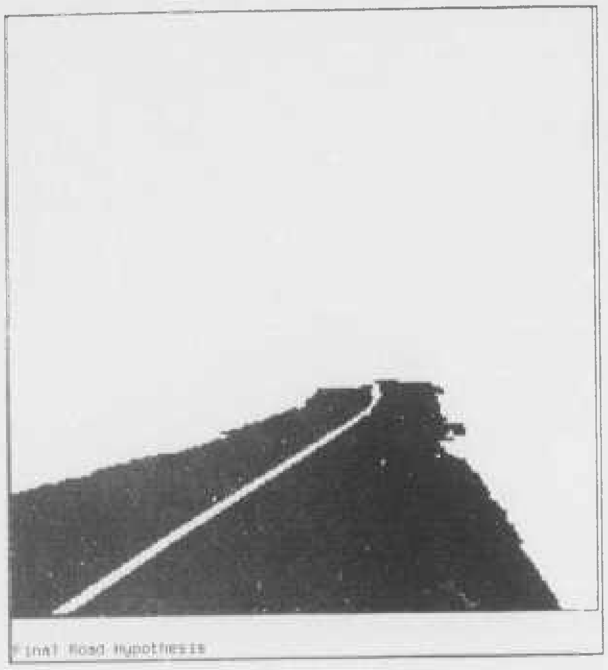
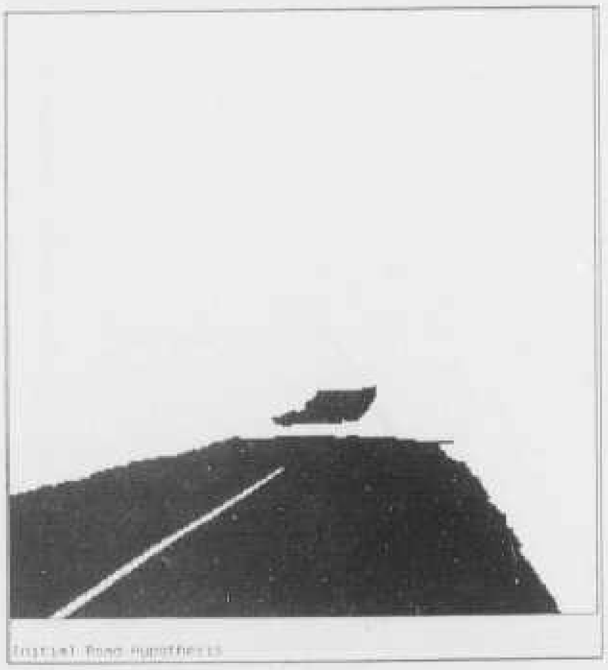


Figure 12a,b. Road hypotheses before and after expansion (a) Initial region and line hypotheses for Figure 2. (b) Final hypothesis.



```
(defvar *phone-pole-description*
(define-object-description
  '((upright-right-edge
      ((spo_related crossbar-bottom-edge
                    crossbar-top-edge)
       (spp_related upright-left-edge)
       (vertical upright-right-edge)
       ((length$>$ 20) upright-right-edge)
       (right-of upright-left-edge)))
    (upright-left-edge
      ((spo_related crossbar-top-edge
                    crossbar-bottom-edge)
       (spp_related upright-right-edge)
       (vertical upright-left-edge)
       ((length$>$ 20) upright-left-edge)))
    (crossbar-top-edge
      (((atop .25) upright-right-edge
                   upright-left-edge)
       (spo_related upright-right-edge)
       (spo_related upright-left-edge)
       (spp_related crossbar-bottom-edge)
       ((atop 0) crossbar-bottom-edge)))
    (crossbar-bottom-edge
      (((atop .25) upright-right-edge
                   upright-left-edge)
       (spo_related upright-right-edge)
       (spo_related upright-left-edge)
       (spp_related crossbar-top-edge)))
    )))
```

Figure 13. Telephone pole object description. In the code the name of each description node is followed by an association list in which the first item of each association pair is the name of a constraint and the second item is a list of nodes, each of which is related to the description node by the named constraint. The constraints spo_related and spp_related are for spatially proximate orthogonal and parallel pairs of lines. (atop .25) is a relation in which one line falls somewhere in the top one-quarter of another line A constraint label like (atop .25) has a function name as the first element; the other elements allow the constraint to be parameterized by supplying extra arguments to the function. The (length> 20) restricts objects of interest to be in the foreground.

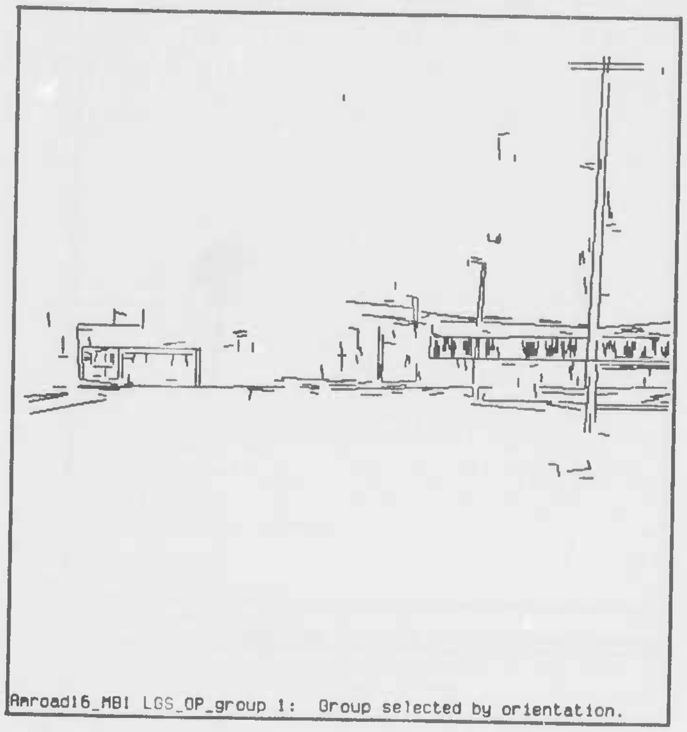
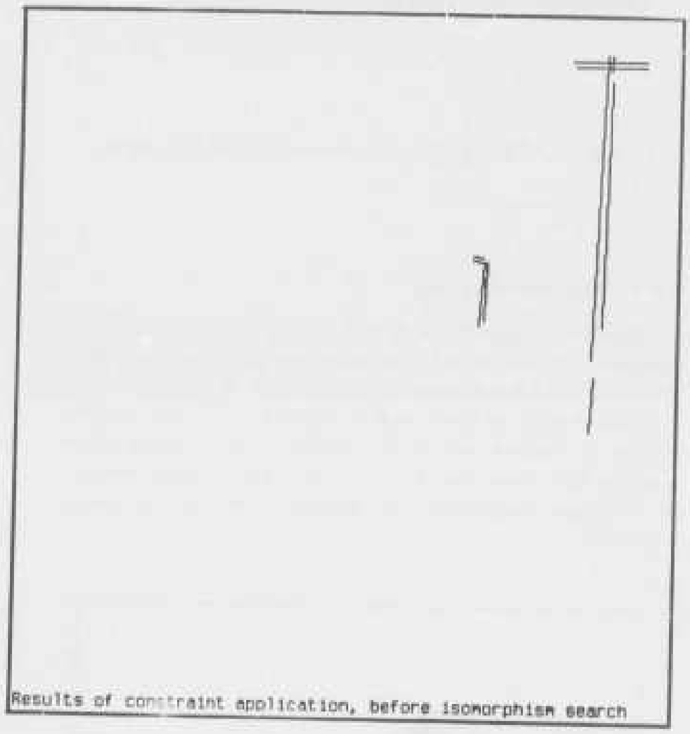
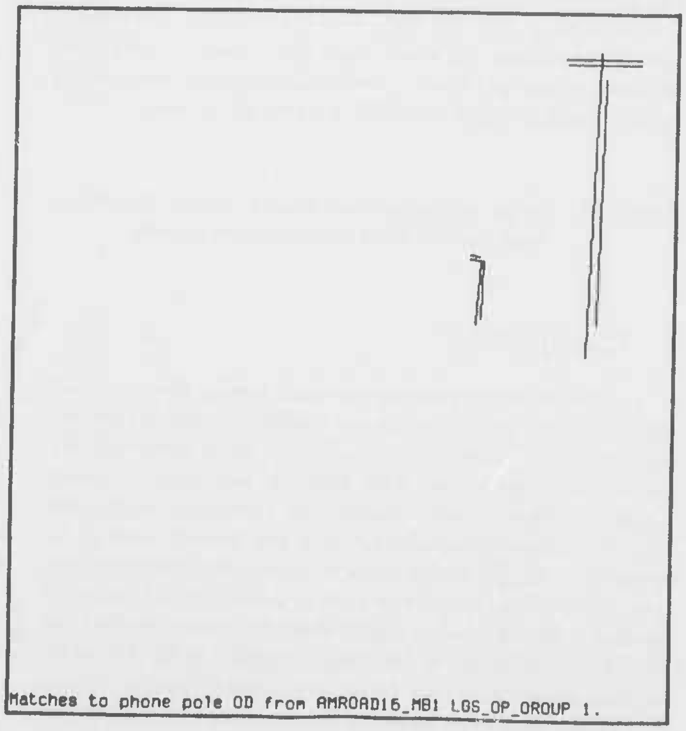


Figure 14a,b,c. Stages in telephone pole object description match.


Results of constraint application, before isomorphism search



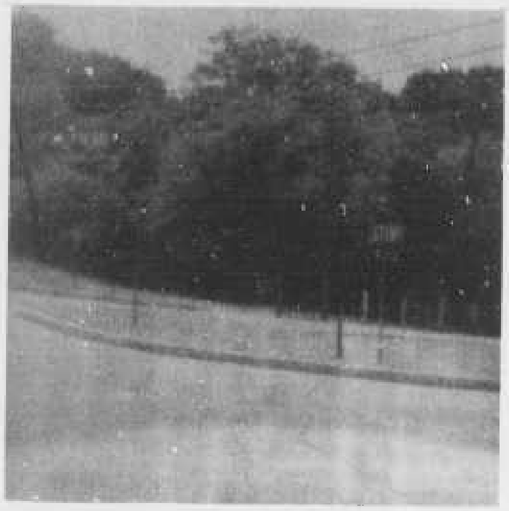


Figure 15. Photograph of intersection with stop signs.

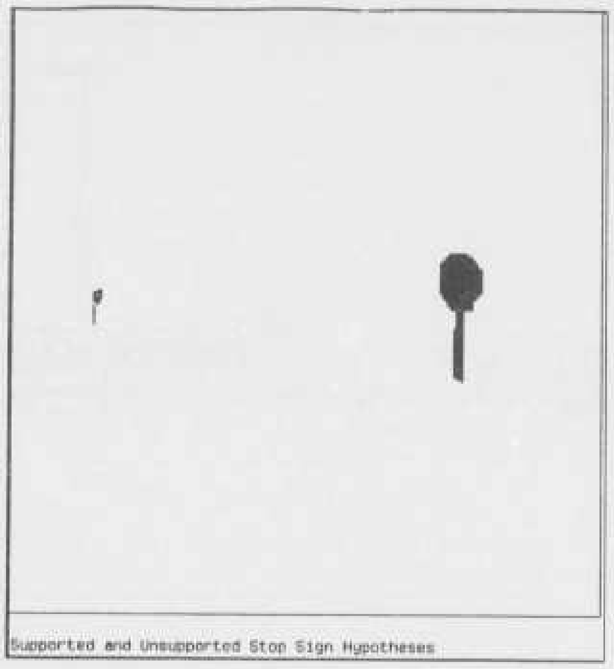
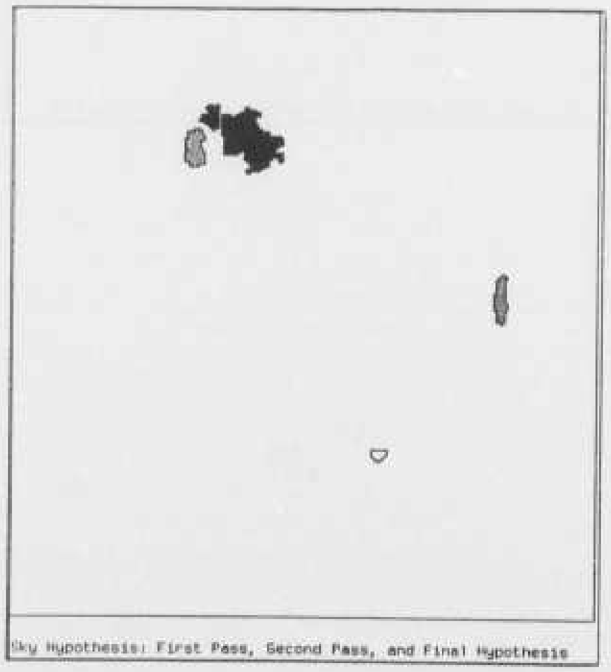


Figure 16. One supported (black) and one unsupported (grey) stop sign hypothesis.

3.2.4 Recognizing Sky

The initial hypothesis strategy for sky is configured as a generator. It initiates a few internal hypotheses on the basis of color and then monitors their progress. If at any point these hypotheses have all been refuted, the strategy relaxes its criteria and generates a new set of hypotheses. This continues until a strong hypothesis for sky is found or the strategy runs out of reasonable candidates. An example of this can be seen in Figure 17.


Sky Hypothesis: First Pass, Second Pass, and Final Hypothesis

3.3 Experimental Environment

The Schema System runs on a TI Explorer lisp machine. The ISR uses a VAX 11/750 (connected to the Explorer by a Chaosnet) as a host for its database. Once the image has been segmented and its lines extracted and grouped, the interpretation process takes about 50 minutes. Much of this time is spent communicating with the VAX over the chaosnet; the simulation of parallelism also slows down the system. A subjective estimate is that an efficient, production implementation of the system could interpret an image in under 10 minutes.

Figure 17. Initial sky hypothesis (clear); relaxed hypothesis (grey and black); final hypothesis (black).

4 Conclusion

A knowledge based computer vision system that produces effective image interpretations on complex natural scenes has been developed. There are several directions of research being pursued with this system. This first is to port it onto a newly acquired multiprocessor. Second, the knowledge engineering tools for implementing schemas in a new domain need to be extended. The goal of this research is for a small development team to be able to implement a set of schemas of the complexity shown here in a week, rather than the month required for this effort. We intend to test this capability in the future by building a knowledge base for an aerial image domain. Third, the current knowledge bases for house scenes and road scenes need to be further developed and generalized, including object schemas which are sensitive to image resolution and object distance. Most importantly, the system must be expanded to use three-dimensional reasoning to create three-dimensional object hypotheses.

5 Acknowledgements

The writers wish to thank a number of people for their help: George Reynolds and Ross Beveridge for rectilinear line grouping, Rob Belknap and Rick Latty for rule development, Charlie Kohl for knowledge-directed token extraction, Nancy Lehrer for automatic rule acquisition, Michael Boldt for line extraction, Rich Weiss, Kelly Murray and Arnie Rosenberg for advice on the object verification knowledge source.

References

[Arbi78] M.A. Arbib, "Segmentation, Schemas, and Cooperative Computation", in *Studies in Mathematical Biology, Part 1*, S. Levin, ed., MAA Studies in Mathematics, Vol. 15, 1978, pp. 118-155.

[Arbi81] M.A. Arbib, "Perceptual structures and distributed motor control", in *Handbook of Physiology: the nervous system, II Motor control*, V.B. Brooks (Ed.), Amer. Physiol. Soc., Bethesda, MD., pp. 1449-1480.

[Belk85] R. Belknap, E. Riseman, and A. Hanson, "The Information Fusion Problem and Rule-Based Hypotheses Applied To Complex Aggregations of Image Events," *Proc. DARPA IU Workshop,* Miami Beach, FL, December 1985.

[Drap86] B. Draper, A. Hanson, and E. Riseman, "A Software Environment for High Level Vision," COINS Technical Report, University of Massachusetts at Amherst, 1986, [in preparation].

[Hans78] Hanson, Allen R. & Riseman, Edward M., "VISIONS: A Computer System for Interpreting Scenes", in *Computer Vision Systems*, Hanson & Riseman, eds., Academic Press, N.Y., 1978, pp. 303-333.

[Kohl87a] C. Kohl, Ph.D. Dissertation, Deparment of Computer and Information Science, University of Massachusetts at Amherst, in preparation, 1987.

[Kohl87b] C. Kohl, A. Hanson, and E. Riseman, "Goal-Directed Control of Low-Level Processes for Image Interpretation," *Proc. of the Darpa Image Understanding Workshop,* Los Angeles, CA, February 1987.

[Lehr87] N. Lehrer, G. Reynolds, and J. Griffith, "Initial Hypothesis Formation in Image Understanding Using an Automatically Generated Knowledge Base," *Proc. of the Darpa Image Understanding Workshop,* Los Angeles, CA, February 1987.

[Reyn87] G. Reynolds and J.R. Beveridge, "Searching for Geometric Structure in Images of Natural Scenes," *Proc. of the Darpa Image Understanding Workshop,* Los Angeles, CA, February 1987.

[Ullm76] J.R. Ullman, "An algorithm for subgraph isomorphism," *Journal Assoc. Comput. Mach.,* Vol. 23, 1976, pp. 31-42.

[Weis86] R. Weiss and M. Boldt, "Geometric Grouping Applied to Straight Lines", *Proceedings on the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Miami, FL, June, 1986, pp. 489-495.

[Weym86] T.E. Weymouth, "Using Object Descriptions in a Schema Network For Machine Vision," Ph.D. Dissertation, Computer and Information Science Department, University of Massachusetts at Amherst. Also, COINS Technical Report 86-24, University of Massachusetts at Amherst, 1986.

Models of Errors and Mistakes in Machine Perception

Part 1. First Results for Computer Vision Range Measurements

Ruzena Bajcsy *Eric Krotkov* *Max Mintz*

GRASP Laboratory
Department of Computer and Information Science/D2
University of Pennsylvania
Philadelphia PA 19104

ABSTRACT

The paper is a first step toward integrating sensor measurements of distance. Its major contribution is to identify and present qualitative models for the errors and mistakes introduced in three particular computer vision distance measurements: range from focus, range from point-based stereo, and range from line-based stereo.

These range measurement techniques are presented as computations, and their dominant sources of error are analyzed qualitatively. We propose to quantify the underlying models for these three range estimation techniques by deriving approximate confidence procedures for the intrinsic parameters and functions which characterize each technique.

1. Introduction and Motivation

When one deals with measurements of any kind, it is always the case that this measurement is accompanied by some error. What is an error? According to the methodology prevalent in physics, chemistry, and other sciences, it is a deviation from some ideal, or some standard. What is an ideal or a standard? The ideal comes from a theoretical description of the phenomenon which invariably is put into some mathematical language. This mathematical representation is also called a model. The word "model" is chosen not accidentally. Indeed it means that it is an idealization of the physical reality under precise conditions. A standard, on the other hand, is usually a very precise physical realization of the ideal with very small error. A standard is the best is some sense that we can get in a physical realization of the model. Error is different from discrepancy. The latter is the difference between two measured values of a quantity, for example obtained by two different measuring devices. There is still a further classification of errors: systematic error and random or stochastic error. Both these errors if well understood can be and should be included into the model, naturally using different mathematical tools for the systematic and stochastic errors. These tools are partly the subject of this paper. Both these errors can be evaluated by calibration of the instruments against standards. While the small random error is related to precision, the small systematic error corresponds to high accuracy.

Now that we have explained what we mean by error, we need to clarify what is a mistake? In our view a mistake is not a large error but a failure of a device or an identifiable component of a device. We realize that there is a very tenuous division line between large errors and mistakes. This is analogous to the problem in pattern recognition of variation within a category (errors) and the difference between categories (mistakes). Taking this analogy in to heart, mistakes may be modeled in the decision theoretic framework or represented in some inheritance graph. We shall explore these possibilities later in the paper.

We have been perplexed why these questions or errors and mistakes were not raised before in the vision community. We think that one reason is that in the past most of the results of various image processing and vision algorithms has been creating new images that are graphically displayed and evaluated by human observers. This kind of evaluation is

Acknowledgements: This work was supported in part by NSF/DCR 8410771, Airforce F49620-85-K-0018, DARPA/ONR, ARMY/DAAG-29-84-K-0061, NSF-CER DCR82-19196 A02, by DEC Corp., IBM Corp., and LORD Corp.

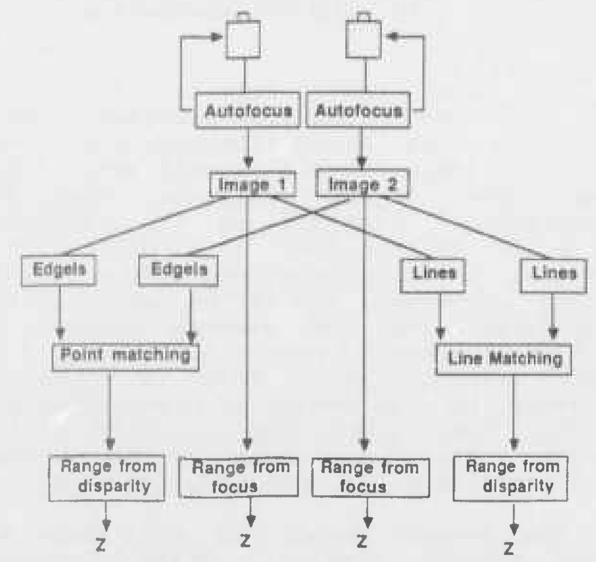
inadequate from many real applications. For example, in industrial inspection, where the visual sensor acts as a measuring device or a probe and the results are used for making decisions on the quality control of the product, the accuracy, precision and error of these measurements become an important issue. Another domain where visual measurements must be used in a feedback is in robotic applications, both in manipulation and mobile robots. Simply, in all cases where a decision is based on visual measurement, it is obvious that the goodness and reliability of these measurements becomes the central issue.

Assumptions and the domain

We shall explore our ideas and mistakes in the domain of computer vision with one particular task: Measurement of the three-dimensional distance. The assumption is that we do not know a priori the distance. We also assume that we have a pair of cameras observing one static, indoor, well-illuminated scene. We shall consider three different approaches of obtaining the distance through vision, as illustrated in Figure 1.

1. Distance from focus;
2. Distance from stereo, using a point matcher;
3. Distance from stereo, using a line matcher.

Figure 1. Architecture of range computations.



There are two reasons why we have chosen to discuss these three particular methods for measuring distance. First, we have implemented and are trying to evaluate them, so we are beginning to understand their theoretical and practical capabilities and limitations. Second, they can be viewed as processes which provide complementary and redundant measurements. Two measurements are complementary (independent) if they measure the same physical quantity with different process; here, focus and stereo provide complementary measurements of distance. Two measurements are redundant if they measure the same physical quantity with the same process; here point-based and line-based stereo provide redundant measurements of distance because they use the same process (triangulation) with different features (points and lines). This is important because if these measurments can be integrated into one best estimate of distance, we will have a basis for integrating measurements from any process, whether complimentary or redundant.

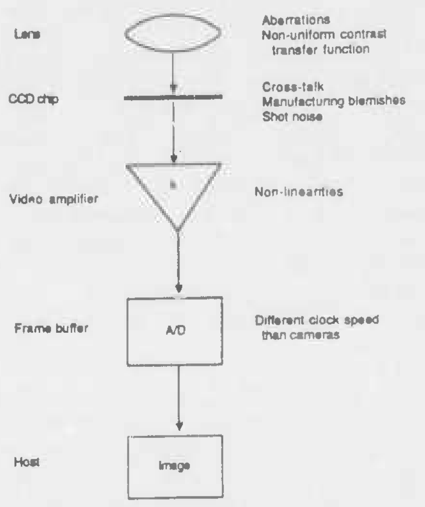
This paper is a first step toward a methodology for integrating sensor measurments using specific sensor models. In our view, its major contribution is to identify and present preliminary models for the errors and mistakes introducted in three particular distance measurements, and to begin thinking about how to combine them. Future work will address quantitative models of errors and mistakes in computer vision distance measurements, and tactile sensing.

This paper consists of eight sections. Section 2 discusses the errors and mistakes introduced in the image formation process. Section 3 describes and analyzes the range from focus technique. Section 4 discusses the computation of distance from stereo disparities, and some errors that apply to all such computations. Section 5 presents and analyzes the point-based stereo technique. Section 6 presents and analyzes the computation of stereo disparities based on finding and matching lines. Section 7 presents a framework for error analysis using confidence procedures. Section 8 ends the paper with general remarks, final conclusions, and the work to be done in the future.

2. Image formation

To the extent that all the distance computations compute local first derivatives of the intensity function, any noise in the image will propagate and be amplified. Hence it is important to understand and model the noise in the digitized values. A good review of the noise characteristics of CCD transducers can be found in the article by Purll [10]. This section will discuss our first results in modeling the noise in our camera system. A more detailed analysis is being prepared by Krotkov, McKendall and Mintz [8]. A crude model of the image formation process is illustrated in Figure 2, which lists some salient features of each component.

Figure 2. Image formation.



Notation

We will denote the image intensity function as the three-dimensional function I, with spatial arguments u and v, and temporal argument t. Since the image intensity is represented as 8 bits, $0 \leq I(u,v,t) \leq 255$.

Much of the analysis involves taking a time series of images. Let $\overline{I}(u,v)$ denote the sample mean of the image intensities over N time samples:

$$\overline{I}(u,v) = \frac{1}{N}\sum_{t=1}^{N} I(u,v,t) . \qquad (1)$$

The spatial variance in a 5x5 neighborhood of the means is computed by:

$$s^2(u,v)=\sum_{i=-2}^{2}\sum_{j=-2}^{2}(\overline{I}(u+i,v+j)-\overline{I}(u,v))^2 . \qquad (2)$$

2.1 Spatial noise analysis

Experiments were conducted to determine the spatial characteristics of the digitization process. The experiments reveal the "signature" of the digitizer with no illumination, with constant illumination, and with illumination from a room scene.

Dark signature

To identify the dark signature of each camera a sequence of N=100 images is taken with the lens cap on. The sample mean over time $\overline{I}(u,v)$ of each pixel $0 \leq u,v \leq 511$ is computed using Equation (1).

For each camera, there are a small number of pixels with non-zero mean and nonzero variance. Repetition of the experiment reveals that the non-zero values occur at the same locations in the respective images. These non-zero values are probably caused by the fact that the CCD elements being digitized into those locations are "blemished". The manufacturer allows a certain number of blemishes on each CCD chip. An element is considered blemished if it exhibits a spurious output (in comparison to its nearest neighbors) of more than 10% of the saturation voltage [6, p. 72]. The blemishes are caused by material variations across the chip, variations in element area, and processing variations in manufacturing (the way silicon is grown). The dark signal, electric current formed by thermal leakage which is indistinguishable (by the video amplifiers) from photocurrent, also carries element-to-element non-uniformities but these are of small magnitude and are effectively "averaged out" by computing $\overline{I}$ [10].

Uniform illumination signature

Next a sequence of images is taken with a uniform illumination, accomplished by placing a nylon diffuser directly over the lens, essentially employing a translucent lens cap. The mean and variance are computed according to Equations (1) and (2), respectively, with N=100.

The first result from this experiment is that the intensities recorded along a 20-pixel band along the exterior border of the image have a significantly lower mean and higher variance than the intensities recorded in the interior. This is an artifact of (1) digitizing a 380x488 array of detectors into a 512x512 array of values, and (2) the fact that the digitizer and the cameras operate at different sample rates, 4.77 MHz an 7.16 MHz, respectively. This is clearly a mistake in the design of our system, and rather than model the behavior of the border values, we choose to ignore them, and use pixels only from the interior.

The second result is that there are some "stuck" pixels. These have large variances, and are located in the same places that were marked as "blemished" above.

The third result from this experiment is that the intensities form an annular pattern, with pixels at a given distance from the center (lying on an ellipse) having equal intensities, and with pixels at greater distances from the center having lesser intensities. This suggests that on-axis light rays are attenuated less than the off-axis rays, i.e., the contrast transfer function of the lens is maximum at its center and diminishes with distance from the center. The intensities seem to vary smoothly inside this annular distribution, and it appears that they vary linearly with field angle (distance to the center of the lens).

2.2 Temporal noise analysis

A number of experiments are being conducted to identify and model the temporal noise characteristics of the digitization process [8]. It is clear that the intensity samples are not temporally independent, i.e., the noise process is characterized by time dependency. This is a significant finding, which profoundly affects the filtering necessary to reduce the noise. We are still investigating the quantitative characteristics of the noise process, and its physical origins.

2.3 Summary

A model of the image formation is a necessity for any further consideration of errors and mistakes, since its parameters come to play in the algorithms for focus, stereo and all other vision and image processing algorithms. Our approach is to treat the whole camera system as a black box and make hundreds of input/output measurements and develop a stochastic model of its behavior. This methodology is especially appropriate when the system is too complicated or not well-enough understood to model its physical behavior explicitly. We feel however that predictive modeling must be used whenever it is possible, since this is the only way to obtain absolute benchmarks, and hence assert absolute mistakes. Having the black box approach only one can speak only about mistakes in a statistical sense.

The conclusions we can draw about the camera system so far are: (1)There are blemished pixels along the border and scattered around the interior of the image, (2)the lens attenuates the intensity of off-axis rays; (3)the intensity samples are not temporally independent. It remains to develop quantitative models of these phenomena.

3. Range from Focus

This section will describe a method for computing range from focus, as well as its inherent and practical limitations. The ideas and results presented here are reported in considerably greater detail by Krotkov [9].

3.1 Method

Our method for computing range from focus involves two steps. First, given the projection $P' = (u,v)$ onto the image plane of an object point $P=(X,Y,Z)$ (Z unknown), we find the lens focal length f which brings P into sharpest focus. Second, given f from step one, we compute the Z-component of P using the thick lens law of first-order optics.

The first problem, how to best determine the focal length providing the sharpest focus on an object point at an unknown distance, is decomposed into two parts: (i)how to measure the sharpness of focus with a criterion function, and (ii) how to optimally locate the mode of the criterion function. The criterion function, which approximates the magnitude of the intensity gradient, can be stated as:

$$\sum_{x,y} S(x,y)$$

where

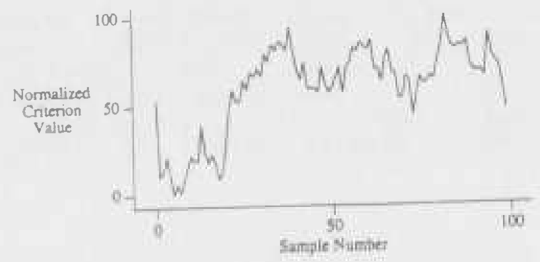
$$S(x,y) = \sqrt{(i_x * I(x,y))^2 + (i_y * I(x,y))^2};$$

$$i_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}; \quad i_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix};$$

* denotes convolution; and x and y range over a (small) neighborhood around P'. In practice this criterion function takes its maximum value when the image is sharply focused, proves to be unimodal, varies monotonically with focal length on either side of the mode, and is relatively easily computed. However it is fairly sensitive to noise in the digitized samples, as illustrated in Figure 3. This problem is solved presently by averaging over many samples.

The Fibonacci search technique is employed to locate the mode of the criterion function, since it is the optimal method for finding the extrema in a unimodal function (the focus criterion function) of one variable (focal length). In general, the Fibonacci search successively narrows the search interval until its size is a given fraction of the initial search region. In practice the Fibonacci search strategy performs

Figure 3. Criterion function temporal variations.



The intensities in a 20x20 window were sampled at 30 Hz, with no filtering. The criterion function was then evaluated for each sample.

extremely well, providing images that appear sharply focused to observers after approximately 11 iterations.

The second problem, how to compute the distance to an object point given the focal length of sharpest focus, is solved by application of the Gaussian thick lens law. The object distance Z is computed as

$$Z = \frac{k(f+a)}{k-f-a} - b \qquad (4)$$

where f is the focal length providing sharpest focus on P′ , and k,a,b, are empirically determined constants.

Using Fibonacci search for the peak of the criterion function and Equation (4), the distance of objects between 1 and 4 meters can be computed with an error less than 5 percent of the object distance. The results for objects at greater distances are not yet available.

3.2 Error analysis

There are a number of limitations of this method. Perhaps the most significant is that it applies to only one point at a time. Thus with a single camera, range from focus can not be computed in parallel. Other limitations include the precision to which d_{in} and f can be measured, the performance of the criterion function, and the nonlinearity of Equation (4).

Spatial quantization is one limitation of this method which can not be circumvented by more precise measurements or better equipment or algorithms. Because the photoreceptors have finite area, an object point may lie at a number of different distances and still be imaged sharply on the same receptor. The distance in object space between the nearest plane and the farthest plane at which satisfactory definition is obtained is the depth of field.

$$\frac{2Zafc(Z-f)}{a^2f^2 - c^2(Z-f)^2} \qquad (5)$$

where f is the focal length, c is the largest dimension of the photoreceptor cell, a is aperture diameter, and Z is the object distance. From Equation (5), it is clear that as c increases, so does the depth of field. The accuracy of the range computation is actually less than the depth of field of the lens, i.e., the computation is as accurate as physically possible. This shows that the dominant source or error in range from focus is the dimension of the photoreceptor cell.

It is a mistake to try to focus on an inappropriate window, one which contains no features or the projection of a depth discontinuity or an occluding edge.

4. Computing range from stereo disparities

There are some errors encountered by any computation of range based on finding disparities. First we will present a method for computing range, and then discuss the important error parameters.

4.1 Method

This section describes the transform from relative disparities (distances in image space) into absolute distances (distances in E^3), based on the geometry illustrated in Figure 4 (loosely based on the analysis by Torre et al [13]). We do not assume that the cameras are parallel.

If (u_l, v_l) are the coordinates in the left image of the perspective projection of an object point P=(X,Y,Z), then by similar triangles:

$$u_l = f\frac{x_l}{z_l+f} \quad \text{and} \quad v_l = f\frac{y_l}{z_l+f} \qquad (6)$$

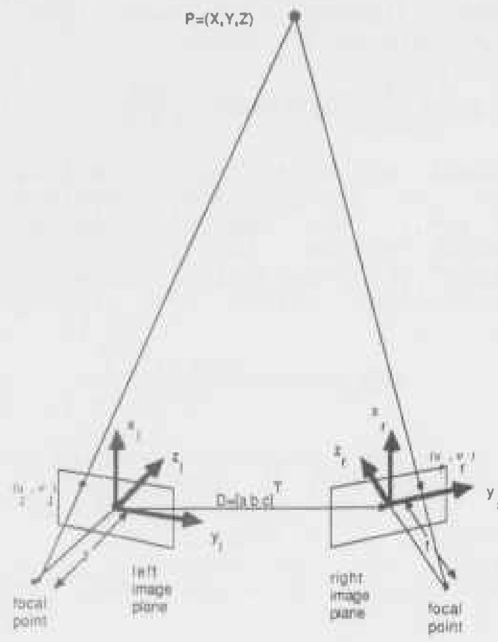
Similarly for the right image:

$$u_r = f\frac{x_r}{z_r+f} \quad \text{and} \quad v_r = f\frac{y_r}{z_r+f} \qquad (7)$$

The two image coordinate systems are related by a translation $\underline{D}=[a\ b\ c]^T$ and a 3x3 rotation matrix [R] (determined by the three Euler angles Φ, θ, Ψ under the $z\,x'\,y''$ convention [4, p.108] such that

$$\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = \begin{bmatrix} R \end{bmatrix} \begin{bmatrix} x_l - a \\ y_l - b \\ z_l - c \end{bmatrix} \qquad (8)$$

Figure 4. Stereo geometry.



The geometry underlying the transformation from relative disparities to absolute distances

Assuming that the scan lines are registered, a=0. Assuming that the optic axes are coplanar (i.e., the cameras are not tilted with respect to each other), Ψ =0. Since our cameras can not roll (rotate around the z-axis), Φ =0.

Under these assumptions Equation (8) is given by

$$\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -sin\,\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_l-a \\ y_l-b \\ z_l-c \end{bmatrix} \qquad (9)$$

Since the matching procedures produce horizontal disparities, we want to express z_l in terms of v_l and v_r . Using the equations in v_l and v_r derived by similar triangles, and solving algebraically for the object distance $Z=z_l$ yields

$$Z = \frac{f}{\xi}\left[(b-v_l)(v_r\sin\theta+f\cos\theta)+f v_r+c\,(f\sin\theta-v_r\cos\theta)\,;\right] \qquad (10)$$

where f is the focal length of the lens, θ is the convergence angle, and $\xi=\sin\theta(v_l v_r+f^2)+f\cos\theta(v_l-v_r)$. For c=θ=0 (orthographic projection) this reduces to the familiar

$$Z = f\,\frac{b-v_l}{v_l-v_r} \qquad (11)$$

4.2 Error analysis

The accuracy of the matcher and the registration procedure limit the accuracy of the computed disparities; this will be discussed in the sections on the matcher error analysis. For given disparities, the range computation is limited by the precision in measuring b, θ, c, and f . Torre et. al. [13] analyzed range errors due to mechanical positioning errors of the cameras, the focal length of lenses, as well as the errors due to lens distortion, in matching and quantization in the picture.

Another error in the distance measurement is due to the fact that an area is projected into one point. This area naturally is bigger with the increased distance from the camera. This last error has been studied by Solina [12], who considered the worst case,and showed that the relative range error ΔZ for focal length f is:

$$\Delta Z = \frac{aZ}{(bf-aZ)} \qquad (12)$$

where a is the pixel size, Z is the object distance, and b is the distance between the focal points (the baseline). From this analysis it follows that a small pixel size is most desirable. This however is quite costly. Another two parameters can be controlled, that is to use longer focal length and/or increase the baseline. Unfortunately both of these parameters if not chosen carefully have bad consequences: too long a focal length decreases the sharpness of focus and too long a baseline makes the matching problem more difficult if not impossible.

5. Point-based stereo

This section will describe a method for computing stereo disparities by matching edge points, as well as its inherent and practical limitations. The ideas and results presented here are drawn from Smitley [11]. The matcher is based on the following assumptions: (1)The two images are properly aligned so that the scan lines correspond to the epipolar lines and hence one can reduce the stereo matching problem into a one-dimensional problem. (2)Matching occurs only there where edge point features exist. (3)Edges occurring on horizontal lines cannot be matched, they are ambiguous. (4)There is an a priori range limit on disparities. All disparities outside of this range are rejected.

5.1 Method

First a registration technique developed by Izaguirre [7] brings the epipolar and scan lines into coincidence. The matching algorithm begins by locating edgels in each image I at multiple resolutions. For a Gaussian window G with a standard deviation σ, let the edge detection filter $f(x,y) = \nabla G(x,y)$. Let P(x,y) denote the output of the edge detection filter at (x,y). P(x,y) is defined by the convolution of $f(x,y)$ with I(x,y), i.e., $P(x,y) = f * I(x,y)$, which has magnitude $\| P(x,y) \| = \sqrt{P_X{}^2 + P_Y{}^2}$ and direction $\theta = \tan^{-1}\left[\frac{P_y}{P_x}\right]$. Edgels are local maxima in $||P||$, computed using Canny's [3] method of non-maximum suppression.

Matching proceeds from coarse (blurred by G with larger σ) to fine resolution. the search window is determined by the size of the filter applied before the edge detection process begins. A left edgel at L= (u_l, v_l) matches a right edgel at R= (u_r, v_r) provided that their gradient vectors are similar ($||P_L|| \approx ||P_R||$ and $\theta_L \approx \theta_R$) and that the intensity distributions around L and R are correlated. Matches from coarser resolutions are used to guide the search for matches at finer resolutions. The disparity of a match (L,R) is the horizontal pixel distance $|v_l - v_r|$.

5.2 Error analysis

The matcher has been tested on diverse images, and generally matches between 75 and 90 percent of the vertically oriented edgels. It has proven to be robust in the presence of noise and poor illumination. However, the matcher commits both errors and mistakes.

Matching errors, i.e., errors in the disparity values, are caused by inadequate localization of edge points. In the absence of noise the localization uncertainty is equal to σ. Hence the smaller the σ, the more sensitive is the edge detector. To model the localization error in the presence of noise, we extend the analysis presented by Bajcsy, Liebman and Mintz [1] to two dimensions, and express the response of the edge filter to an intensity function I(x,y) which is the sum of a noise-free image $I_0(x,y)$ and a zero mean white noise process v(x,y) with constant spectral density r. The response of the edge filter to the noise process v(x,y) is again a second-order wide-sense stationary stochastic process with zero mean and variance ρ, where

$$\rho = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} r\, f^2(x,y)\,dxdy = \frac{\sqrt{2}r}{8\pi\sigma^4} \qquad (13)$$

The effect of noise on estimating the location of (x*,y*)--the maximum of $P_0(x,y)$, i.e., the maximum response of the edge filter to $I_0(x,y)$--can be determined by considering the effective signal-to-noise ratio SNR at point (x*,y*):

$$SNR(x^*,y^*) = \frac{P(x^*,y^*)}{\sqrt{\rho}} \qquad (14)$$

For a unit ramp intensity function in one dimension, SNR (x*) is proportional to $\sqrt{\sigma/r}$. This result is not significantly different in two dimensions, and the important point is that the signal to noise ratio depends directly upon σ.

From the analysis above, it follows that the disparity error due to point-matching of an edge element pair is a function of σ. Without noise, the disparity error is $\pm\sigma$. With noise the disparity error is proportional to $\sqrt{\sigma/r}$. In summary, the error of the whole process is the sum of all the errors from the cameras and point localization.

Since the disparity errors are a function of point localization, we propose that the point matching process itself results not in errors but in mistakes only. A mistake, i.e., a false match, occurs only when one of the four stated assumptions does not hold. This kind of mistake is analagous to the failure of a physical device.

6. Line-based stereo

This section will describe a method for computing stereo disparities based on matching lines, as well as its inherent and practical limitations. The ideas and results presented here are drawn from the forthcoming thesis by Henriksen [5].

6.1 Method

There are two basic steps in this stereo approach, and hence two different error sources: line extraction and line matching to compute stereo disparities.

The line extraction procedure goes as follows:

1. Compute the gradient of the grey-value function at every pixel in the

picture.

2. Group pixels into edge-support regions based on similarity of gradient orientation.

3. Approximate the grey-value function in an edge support region by plane.

4. Compute a weighted average of the grey-values in the edge-support region, and use it to determine a horizontal plane.

5. Intersect the two planes computed in step 3 and step 4, giving an infinitely long straight line.

6. Project the line computed in step 5 onto the picture plane, and intersect it with the boundaries of the edge-support region, giving a line segment.

This algorithm produces a polygonal description of a picture, i.e., the picture is described by a set of line segments.

Line matching goes as follows:

1. Construct an Adjacency Graph such that: (a) for a given line it can tell which are neighbors to the given line, and (b) for a given point it cal tell which lines are in the neighborhood of the given point.

2. Construct a Disparity Graph providing similar information as the Adjaceny Graph but in a smaller neighborhood.

3. Generate Hypotheses, in four steps:

    1. Select a line segment L (let us say in the left) image.
    2. Compute the midpoint of the line segment.
    3. Compute the epipolar line in the right image corresponding to the midpoint of L.
    4. Find all line segments R in the right image which satisfy the following constraints:
        a) R intersects the epipolar line corresponding to the midpoint of L
        b) disp(L,R) <disp[max]
        c) length(L)≈length(R)
        d) arg(ΔL) arg(ΔR)
        e) $||L||\approx||R||$

    The set of line pairs {L,R,disp} found under setp 4 is a set of possible matches, and the elements in this set are called the hypotheses derived from L.

4. Verify hypotheses by testing whether the hypothesis satisfies two constraints: (i)Uniqueness - a line segment in one image can only have one match in the other image; (ii)Continuity - the disparity of neighboring line segments must have similar disparities.

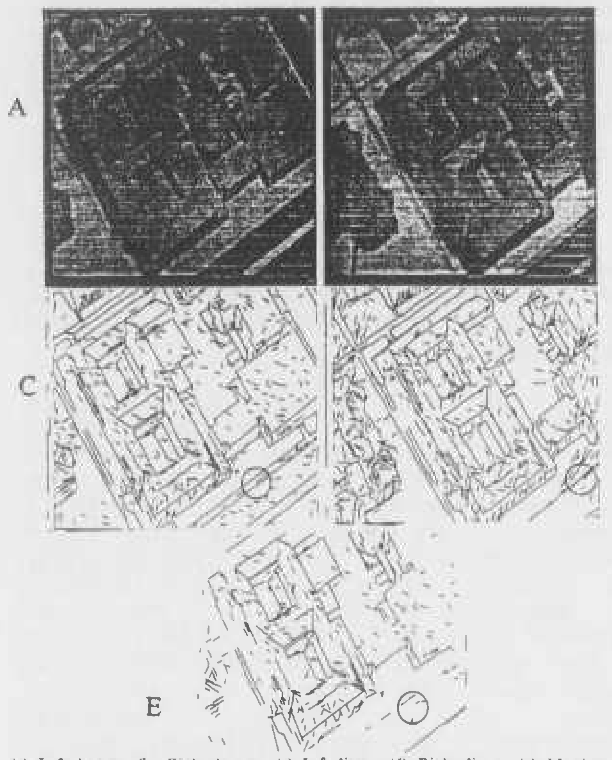
6.2 Error analysis

As in the case of the point matcher, the line matcher commits both errors and mistakes. Because the line matcher uses more local information than the point matcher, its error parameters are more complicated.

In the line extraction process the computation of the gradient causes error , which is exactly the same as the point localization error in the disparities computed by the point matcher. In addition to the point localization error, the grouping of edges into regions is a source of error which has several parameters. The first parameter can be quantified by the size of the edge-support neighborhood. The larger the neighborhood, the larger the danger of including irrelevant information. The second parameter can be quantified by the least-squares residual for fitting the grouped points to a plane. The larger the residual, the worse is the fit to a plane.

In the stereo matching procedure errors, may occur during the hypothesis generation part, where the properties of matched lines must agree up to a certain ϵ measuring the tolerance with which two angles or two lengths must agree. This is not so large. Mistakes occur during the verification part where the uniqueness constraint (that is a line segment in one image can have only one match in the other image) prevents matches of similar lines in the neighborhood. This is illustrated in Figure 5. Figure 5a and 5b show the original left and right images; although the reproductions do not show it so clearly, there is a significant contrast difference in the two images. Figure 5c and 5d show the lines extracted from the original images. In the window marked, the left image contains two parallel lines, and the right image contains only one. This is due to (i)the contrast difference between the two images, and (ii)the fact that the gradient directions in the left image vary little and in the right image they vary greatly. As illustrated in Figure 5e, in this case the system understandably matches only one line instead of two. This is an example of an error (failure of line detection) compounded into a mistake (not matching a line).

Figure 5. Example of line-matching.



(a) Left image. (b) Right image. (c) Left lines. (d) Right lines. (e) Matches.

It is an interesting consequence of this analysis that for more regular objects one needs bigger and bigger features for a unique match.

7. Error Analysis Via Confidence Procedures

We propose to quantify the underlying models for these three range estimation techniques by deriving approximate confidence procedures for the instrinsic parameters and functions which characterize each technique. The confidence measure will provide us with a methodology for: (i) quantifying the individual sources of error based on sensor measurements, (ii) deriving error measures for computed parameters, i.e., the effect of the propagation of the measurement errors, and (iii) obtaining sensitivity models for our measurement techniques and computational algorithms.

7.1 Concepts and Definitions

Confidence procedures can be viewed as set-valued estimates, as opposed to point estimates. In order to compare these concepts, we make use of the following specific example. Suppose $\underline{Z} = (Z_1, Z_2 \ldots, Z_n)^T$ denote a vector of n independent identically distributed IID) observations which are normal with mean θ and variance $\sigma^2 (N(\theta,\sigma^2))$. Assume that σ^2 is known, and that θ is an unknown point in a known parameter set Ω. In this example we will assume that $\Omega =$ [-d,d]. Then, based on this given measurement model, we find: (i) a point estimate for θ based on $\underline{Z}$; and (ii) a fixed size confidence procedure for θ based on $\underline{Z}$.

7.2 Point Estimation

In order to obtain a point estimate for θ , we shall employ the method of maximum likelihood estimation (MLE) (Bickel and Doksum, [2]). The method of maximum likelihood dictates that we compute δ_M ($\underline{Z}$), where:

$$\delta_M(\underline{Z}) = arg_\theta \max f(\underline{Z} \mid \theta) \qquad (15)$$

and f ($\underline{Z} \mid \theta$) denotes the conditional density of $\underline{Z}$ given θ.

The application of the MLE method in this example yields:

$$\delta_M(\underline{Z}) = d, \quad \bar{Z} < d; \qquad (16)$$
$$\delta_M(\underline{Z}) = \bar{Z}, \quad |\bar{Z}| \le d; \qquad (17)$$
$$\delta_M(\underline{Z}) = -d, \quad \bar{Z} < -d; \qquad (18)$$

where $\bar{Z} = \frac{1}{n}\sum_{i=n}^{n} Z_i$.

7.3 Confidence Procedures

The decision rule δ^* ($\underline{Z}$) defines an optimal fixed size confidence procedure of width 2e, if for all δ ($\underline{Z}$):

$$\inf_\theta P_\theta[\theta \epsilon C^*(\underline{Z})] \ge \inf_\theta P_\theta[\theta \epsilon C(\underline{Z})]; \qquad (19)$$

where:

$$C^*(\underline{Z}) = [\delta^*(\underline{Z})-e, \delta^*(\underline{Z})+e]; \text{ and} \qquad (20)$$
$$C(\underline{Z}) = [\delta(\underline{Z})-e, \delta(\underline{Z})+e]. \qquad (21)$$

We illustrate the solution $\delta^*(\underline{Z})$ to this problem for the case where d=3e:

$$\delta^*(\underline{Z}) = 2e, \quad a+2e \le \bar{Z}; \qquad (22)$$
$$\delta^*(\underline{Z}) = \bar{Z}-a, \quad a \le \bar{Z} \le a+2e; \qquad (23)$$
$$\delta^*(\underline{Z}) = 0, \quad -a \le \bar{Z} < a; \qquad (24)$$
$$\delta^*(\underline{Z}) = \bar{Z}+a, \quad -a-2e \le \bar{Z} \le -a; \qquad (25)$$
$$\delta^*(Z) = -2e, \quad \bar{Z} < -a-2e; \qquad (26)$$

where: the parameter a satisfies 2F(-a-e) =F(a-e); and F denotes the CDF of an $N(0,\sigma^2/n)$ variate.

The general solution to this problem appears in [14]. This solution can be extended to include a wide variety of non-Gaussian sampling distributions with uncertain scale factors, as well as ϵ-contaminated sampling distributions. The details of these extensions appear in [15].

8. Discussion

This paper has presented an analysis of the errors and mistakes made in computing the distance of objects using three particular range measurement techniques: focus, point-based stereo, and line-based stereo. First we looked at the noise in the image formation process. then the range measurement techniques were presented in some detail, and their most important error parameters were identified. Finally confidence procedures were proposed to quantify those error parameters.

The noise in the image formation process propagates into all of our computations, so it is very important to model this noise. Our first qualitative findings about this model are (1)there are blemished pixels along the border and scattered around the interior of the image; (2)the lens attenuates the intensity of off-axis rays; (3)the temporal noise is probably not normally distributed. It remains to develop quantitative models of these phenomena.

For the range from focus computation the errors include the precision with which the focal length can be measured, the temporal variations of the criterion function due to digitization noise, and the size of the photoreceptor cells. The latter is the dominant error for our present implementation. Mistakes are committed by trying to focus on a window with no features, or with features at different object distances.

The errors in the range from stereo disparity computation are due to the precision in measuring (i)the focal length of the lens, (ii)the displacement between the cameras, (iii)the convergence angle, and the errors in the disparities computed by each of the matching algorithms. The accuracy of the stereo disparities computed by point-matching is determined by the uncertainty of point localization, which is $\pm\sigma$ No errors are introduced by the matching, only mistakes. The accuracy of the stereo disparities computed by line-matching is determined by the uncertainty of point localization ($\pm\sigma$ and errors in line-finding which can be quantified by the size of the grouping region and the least-squares residual.

In general, the error of a matcher is disproportional to the complexity of the matched feature. So the edge-based matcher will be more error-prone than the line-based matcher. This confirms the intuitive notion that edges are less reliable features than let us say lines or other bigger features. In fact as we said before the matcher makes only mistakes. Hence our final important conclusion about computing range from stereo: One cannot accept the disparity values from stereo without some guidance which can come either from a priori expected knowledge about the distance or from other more direct measurement of the distance, like range from focus or vergence. Then one proceeds: if the range from stereo agrees with some other measurement of the same distance then neighboring distance values computed from stereo are accepted otherwise they are rejected.

In our view, the major contribution of this paper is to identify and present preliminary models for the errors and mistakes introduced in three particular distance measurements, and to begin thinking about how to combine them. Future work will explore the quantitative aspects of the error parameters that have been identified.

1. Bajcsy, R., M. Mintz, and E. Liebman, "A Common Framework for Edge Detection and Region Growing," Submitted to the 9th Intl. Conf. Pattern Recognition, Paris (October, 1986).
2. Bickel, P. J. and K. A. Doksum, Mathematical Statistics, Holden-Day (1977).
3. Canny, J.F., "Finding Edges and Lines in Images," MIT AI-TR-720 (1984).
4. Goldstein, H., Classical Mechanics, Addison-Wesley (1966).
5. Henrikson, K., "Line-based Stereo Matching," University of Pennsylvania TR (1986)
6. Imaging, Fairchild CCD, CCD: The Solid State Imaging Technology, 1980.
7. Izaguirre, A., P. Pu, and J. Summers, "A New Development in Camera Calibration--Calibrating a Pair of Mobile Cameras," IEEE Conference on Robotics and Automation, St. Louis, pp. 74-79 (March, 1985).
8. Krotkov, E.P., R. McKendall, and M. Mintz, "Statistical Models of Camera System Noise," In preparation (1986).
9. Krotkov, E.P., "Focusing," University of Pennsylvania T.R. CIS-86-22 (1986).
10. Purll, D.J., "Solid-state image sensors," in Automated Visual Inspection, ed. B.G. Batchelor, Elsevier Science Publishing Company, New York (1985).
11. Smitley, David, "The Design and Analysis of a Stereo Vision Algorithm," University of Pennsylvania TR MS-CIS-85-27 (1985).
12. Solina, F., "Errors in stereo due to quantization," University of Penn. T.R. (December, 1985).
13. Torre, V., A. Verri, and A. Fiumicelli, "The Stereo Accuracy for Robotics," International Symposium of Robotics Research(3), pp. 89-93 (1985).
14. Zeytinoglu, M. and M. Mintz, "Optimal Fixed Size Confidence Procedures for a Restricted Parameter Space," Ann. Statist. 12, pp. 945-957 (1984).
15. Zeytinoglu, M. and M. Mintz, "Robust Fixed Size Confidence Procedures for a Restricted Parameter Space," Submitted for publication to Ann. Statist. (1985).

Automating Knowledge Acquisition For Aerial Image Interpretation

David M. McKeown, Jr.
Wilson A. Harvey

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, PA. 15213

Abstract

The interpretation of aerial photographs requires knowledge about the scene under consideration. Knowledge about the type of scene: airport, suburban housing development, urban city, can aid in low-level and intermediate level image analysis, and can be expected to drive high-level interpretation by constraining search for plausible consistent scene models. In this paper we describe the organization of a set of tools for interactive knowledge acquisition of scene primitives and spatial constraints for interpretation of aerial imagery. These tools include a user interface for interactive knowledge acquisition, the automated compilation of that knowledge from a schema-based representation into productions that are directly executable by our interpretation system, and a performance analysis tool that generates a critique of the final interpretation. Finally, the utility of these tools is demonstrated by the generation of rules for a new task, suburban house scenes, and the analysis of a set of imagery by our interpretation system.

1. Introduction

In this paper we describe a collection of software tools, ISCAN/RULEGEN/SPATS, for interactive acquisition of spatial knowledge, automated compilation of this knowledge into a rule-based scene interpretation system, and the production of performance analysis statistics to aid in incremental refinement of spatial knowledge. This work is focused on knowledge acquisition and performance analysis tools for SPAM, a knowledge-based system designed to interpret aerial photographs for mapping and photo interpretation. We have reported on SPAM research results in the context of airport scenes[1, 2].

We address a broad set of topics within the overall framework of knowledge acquisition. First and foremost we are interested in automating the process by which an interpretation system, such as SPAM, can assimilate new knowledge to improve performance on existing interpretation tasks, or in attempting to begin to become proficient in new ones. For the airport task we primarily relied on spatial constraints found in books on airport design[3, 4, 5] and, to a lesser extent, by observations of relationships found in aerial imagery. Other task domains, such as suburban house scenes, do not appear to have codified spatial organizations, although they exhibit similar patterns across many examples. In lieu of such information the ability to indicate and measure spatial relationships in representative imagery becomes more important. ISCAN is our first attempt to provide a graphical user interface, appropriate in an image-based domain, which has a model of the types of knowledge required by SPAM during the interpretation process. Such an interface may also provide individuals such as cartographers, remote sensing and photo interpreters, and other non-programmers with a mechanism for adding knowledge to SPAM without a detailed understanding of the underlying system.

A second research goal is to explore the generality of the SPAM architecture for a variety of tasks within the general domain of aerial image interpretation. RULEGEN is a tool that compiles spatial and structural knowledge, stored as collections of rule schemata, and generates productions that are executed by SPAM. RULEGEN was partly motivated by difficulties encountered in extending and generalizing SPAM, which was developed to interpret

* This research was sponsored by the Defense Mapping Agency, under Contract DMA 800-85-C-0009 and by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory Under Contract F33615-81-K-1539. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Mapping Agency, of the Defense Advanced Research Projects Agency, or the US Government.

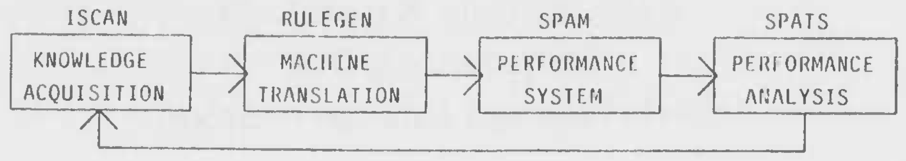


Figure 1-1: Overview of Knowledge Acquisition For SPAM

airport scenes, to interpret simple suburban house scenes. Many of these difficulties impacted our ability to easily add and delete rules to measure the effect of knowledge in various phases of the interpretation process. Changes in the knowledge base often generated unforeseen interactions between the control of rule execution within the interpretation phases. In a system with over 500 productions the management of such changes became a significant burden. As we began to address these issues it became clear that the solution was to view SPAM as an interpretation architecture within which we could embed specific task knowledge. RULEGEN was developed to automatically generate the core task-independent evaluation and control functions that represent the SPAM interpretation architecture and to take task-dependent knowledge in the form of rule schemata and compile productions whose execution was embedded within this core. Therefore, the performance system, SPAM, can be completely generated by RULEGEN when it is supplied with appropriate task-dependent knowledge.

Finally, SPATS was motivated by a need to automate the evaluation of the interpretations produced by SPAM within the context of idealized human photo interpretation. The goal was to measure the size of the interpretation space explored by SPAM, the number of competing hypotheses, and the correctness of those hypotheses during each interpretation phase. By varying the image segmentations presented to SPAM or by generating SPAM systems with different types of spatial knowledge we can now more rigorously evaluate and explore knowledge effects using SPATS. Figure 1 is an abstract overview of the relationship between these tools. While this particular focus on acquisition, compilation, and performance evaluation might appear to be somewhat parochial, we believe that these issues will be seen to be central to other researchers in computer vision working along similar lines.

Section 1.1 briefly outlines some related research and describes our views on knowledge acquisition for computer vision. Section 1.2 gives the layout of the remainder of the paper.

1.1. Knowledge Acquisition For Vision

Previous efforts to investigate knowledge acquisition within the context of systems for image interpretation have primarily focused on spectral properties of objects in the image or viewpoint specific spatial relationships. Early work by Barrow and Popplestone[6] addressed the problem of describing relations between picture elements with predicates like ADJACENT(x,y) or ABOVE(x,y). Using this methodology "rules" could be formulated from these predicates and attached to individual elements of a picture. For example, in the context of face recognition, a nose would be defined by the rule: "ABOVE(x,mouth) and LEFT-OF(x,right-eye) and RIGHT-OF(x,left-eye)". These rules were to be embedded into a resolution theorem proving paradigm. This work was a basis for the ISIS[7] system which added the use of an interactive segmentation system. It allows a user to interactively specify representative regions with a particular interpretation, and then invoked an intensity classification segmentation process to attempt to extract the remaining parts of the scene.

Recently, the VISIONS system[8, 9] has reported similar attempts to make interpretations by propagating low-level process output, such as lines or regions, up to an intermediate level, which combines the low-level output with computed attributes such as color, texture, or orientation. Interpreted objects are defined in terms of these intermediate elements. Loosely speaking these classification systems use "knowledge" such as *the sky has a pixel intensity greater than 30 but less that 125 in the blue band*. In fact, one must resort to density weighting functions much as in statistical pattern recognition for remote sensing. This "knowledge" is highly sensor and scene dependent. Other measures such as height, size (in pixels), and relative spatial position (e.g. *sky is above the house* and *grass is below tne house*) are also employed. Again, these viewpoint dependent quantities will vary, not only from domain to domain, but from image to image. Ultimately *sky is blue* and *grass is green* allows for a direct mapping between regions and the associated high-level interpretation. However, this mapping represents a rather shallow use of knowledge whose robustness is questionable.

For example, consider the effect of averaging the RGB components of a color image into a monochromatic image. While the scene geometry remains unchanged, without the direct mapping of region spectral properties into a semantic interpretation (*sky is blue*) it is difficult to see how to operationalize much of the spatial knowledge. Thus, although there appears to be a spatial component, it is predicated on strong mapping between color and interpretation.

In our work with SPAM we have attempted to identify sources of knowledge that did not suffer from these drawbacks, and utilize spatial relationships in such a way that a chain of reasoning exists, generated from the application of many constraints across multiple levels of interpretation. While spectral knowledge can play a role in certain domains we believe that there are many types of spatial knowledge that can be expected to be more effective in driving the knowledge-based interpretation of aerial imagery. In terms of acquisition and utilization, we believe that Figure 1-2 lists the types of knowledge that are available and appear to us to be effective in aerial image interpretation tasks.

1. Knowledge for the determination and definition of appropriate scene domain primitives. This includes knowledge of the image segmentation process, the image analysis tools that can reliably extract these primitives, and the appearance of the primitives in the image.
2. Knowledge of spatial relationships and constraints between the scene domain primitives.
3. Knowledge of model decompositions that determine collections of primitives which form "natural" components of the scene. These components can be characterized as sub-models that accumulate support for local interpretations and provide a context within which global analysis can be performed.
4. Knowledge of methods for combining these components into complete scene interpretations.
5. Knowledge of how to recognize and evaluate conflicts between competing interpretations.

Figure 1-2: Types Of Knowledge Utilized In SPAM

1.2. Layout of the Remainder of Paper

In the following section we briefly describe the architecture of SPAM. We discuss the kinds of knowledge that SPAM utilizes and therefore needs to be acquired for an interpretation task. In Section 3 we describe the ISCAN/RULEGEN/SPATS tools and in Section 4 give an example of the schemata produced by ISCAN and used by RULEGEN to generate a SPAM interpretation system. Finally, in Section 5 we give an example of suburban house scene interpretation by a SPAM system generated using the ISCAN/RULEGEN/SPATS tools. We also compare the structure of the original hand generated SPAM system with those generated using these knowledge acquisition tools.

2. The SPAM Architecture

SPAM represents four types of interpretation primitives, *regions*, *fragments*, *functional areas*, and *models*. SPAM performs scene interpretation by transforming image *regions* into scene *fragment* interpretations, aggregating these fragments into consistent and compatible collections called *functional areas*, and selecting sets of functional areas that form *models* of the scene. Loosely speaking there are four *phases* of interpretation. Each of these phases operationalizes one type of domain knowledge. In order to build a SPAM system we must be able to acquire knowledge for each interpretation phase.

Phase 1: Region-to-fragment
  Assigns the image region data a set of fragment interpretations based solely on local properties (2-D shape characteristics, texture, 3-D depth/height, etc.) and knowledge about the classes of objects found in the scene.

Phase 2: Local-consistency-check
  Pair-wise tests are performed on the fragment interpretations that utilize spatial knowledge about the scene under consideration. The confidence of those interpretations supporting one another are incremented based on the quality of the test.

Phase 3: Functional-area
  Sets of mutually consistent interpretations that share similar functions or are spatial decompositions of the scene are grouped into cliques called functional areas.

Phase 4: Model-generation
  Sets of functional areas are grouped together into scene segments. The segments with the largest number of functional areas become distinct scene models. Any conflicts encountered when combining functional areas are resolved by a default strategy, using the accumulated support for each interpretation, or by specific knowledge added by the user.

As shown in Figure 2-1 each phase is executed in the order given above. SPAM drives from a local, low-level set of interpretations to a high-level, more global, scene

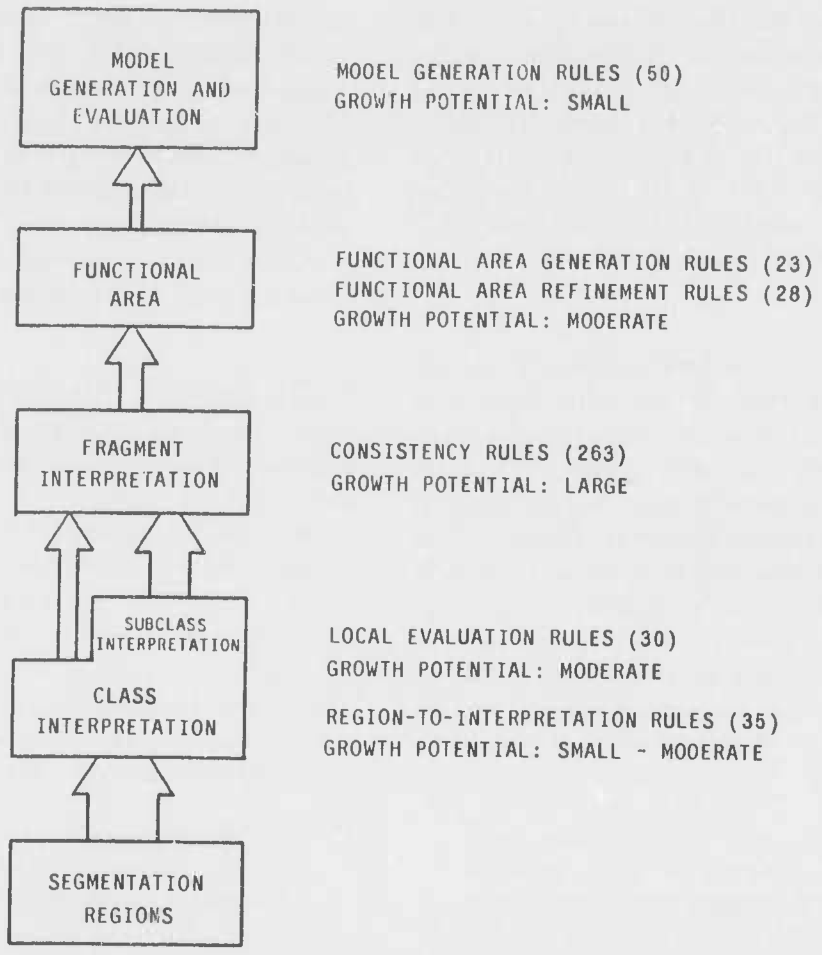

MODEL GENERATION AND EVALUATION
MOOEL GENERATION RULES (50)
GROWTH POTENTIAL: SMALL
FUNCTIONAL AREA
FUNCTIONAL AREA GENERATION RULES (23)
FUNCTIONAL AREA REFINEMENT RULES (28)
GROWTH POTENTIAL: MOOERATE
FRAGMENT INTERPRETATION
CONSISTENCY RULES (263)
GROWTH POTENTIAL: LARGE
SUBCLASS INTERPRETATION
LOCAL EVALUATION RULES (30)
GROWTH POTENTIAL: MODERATE
CLASS INTERPRETATION
REGION-TO-INTERPRETATION RULES (35)
GROWTH POTENTIAL: SMALL - MOOERATE
SEGMENTATION REGIONS

Figure 2-1: Interpretation Phases In SPAM

interpretation. There is a set of hard-wired productions for each phase that control the order of rule executions, the forking of processes, and other domain-independent tasks. However this "bottom-up" organization does not preclude interactions between phases. For example, prediction of a fragment interpretation in *functional-area* phase will automatically cause SPAM to reenter *local-consistency* phase for that fragment. Other forms of top-down activity include stereo verification to disambiguate conflicting hypotheses in *model-generation* phase and linear alignment in *region-to-fragment* phase. Figure 2-2 shows the refinement/consistency/prediction paradigm used in SPAM within each interpretation phase. Knowledge is used to check for consistency among hypotheses, to predict missing components using context, and to create contexts based on collections of consistent hypotheses. Prediction is restrained in SPAM in that hypotheses cannot predict missing components at their own representation level. A collection of hypotheses must combine to create a context from which a prediction can be made. These contexts are refinements or spatial aggregations in the scene. For example, a collection of mutually consistent runways and taxiways might combine to generate a runway functional area. Rules that encode knowledge that runway functional areas often contain grassy areas or tarmac may predict that certain sub-areas within that functional area are good candidates for finding such regions. However, an isolated runway or taxiway hypothesis cannot directly make these predictions. In SPAM the context determines the prediction. This serves to decrease the combinatorics of hypothesis generation and to allow the system to focus on those areas with strong support at each level of the interpretation.

2.1. Knowledge Acquisition In SPAM

In order to automate knowledge acquisition for SPAM we must be able to identify the kind of knowledge required for each of the interpretation phases described in the previous section. In this section we describe this with respect to the 5 types of knowledge defined in Figure 1-2.

The type of knowledge required in *region-to-fragment* phase is the definition of the shape and appearance properties of objects in the task domain, organized as

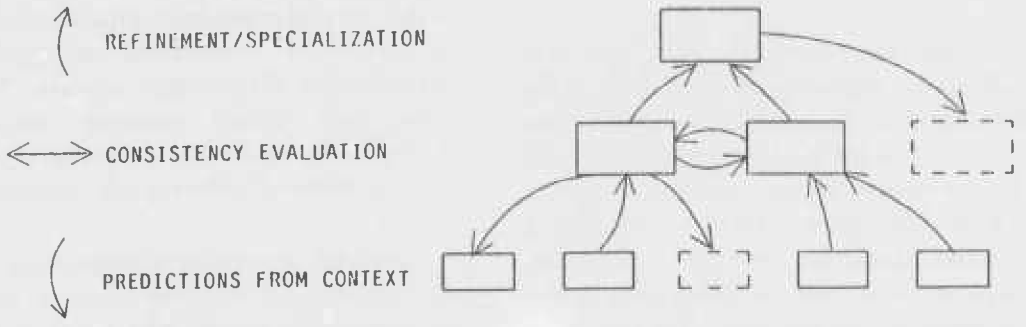


Figure 2-2: Refinement, Consistency, and Prediction in SPAM

coarse classes of similar objects with specializations based on finer intra-class distinctions. For example, in a coarse sense, linear features such as roads, runways, taxiways can be grouped in one class, while hangars, maintenance buildings, control towers, and terminal buildings would be another coarse class in an airport interpretation task. Each of the members of the class can be specialized with constraints such as runways are never curved, while roads may be curved. Heights, sizes, and specific shape criteria might be used to specialize the building class. This type of knowledge is best represented as Type 1 in Figure 1-2.

During *local-consistency-check* phase knowledge of the structure or layout of the task domain, i.e. airports, suburban housing developments, is used to provide spatial constraints for evaluating consistency among fragment hypotheses. Type 2 knowledge is required from the user or other sources. For example, *'runways intersect taxiways'*, and *terminal buildings are adjacent to parking apron'* are the kinds of knowledge in terms of spatial relationships that we would like to capture for an airport interpretation task. It is important to assemble a large collection of such consistency knowledge since these tests are used to assemble fragment hypotheses found to be mutually consistent as contexts for further interpretation.

There are two types of knowledge necessary to perform *functional area* phase. The first is primarily Type 3 knowledge which defines collections of objects that form spatial decompositions within the task domain. For example, knowledge that *runways*, *taxiways*, and the *grassy areas* that separate them from the area where planes takeoff and land can be used as one partition of the overall airport scene. Within this context Type 5 knowledge aids in prediction of missing components, selection of competing hypotheses, or in defining methods for disambiguating conflicting interpretations. For example, 'if a runway functional area has been formed and it contains a terminal building fragment then use stereo verification to confirm or refute that fragment hypothesis.' In other cases knowledge that simply selecting the competing fragment with the highest confidence based upon cumulative application of *region-to-fragment* and *local-consistency-check* rules may be appropriate.

Finally, during *model generation* phase, Type 4 knowledge consisting of how to combine spatial decompositions and Type 5 knowledge consisting of how to recognize and evaluate conflicts that arise during this aggregation must be acquired. However, much of this is simply selecting a strategy, i.e., 'use the functional areas with the highest confidence that have no conflicts', or 'find the maximal set of compatibles regardless of confidence'. The process for performing these alternative combinations is, in some sense, hardwired in the SPAM architecture as a set mutually exclusive methods and only the method is directly specified during knowledge acquisition. In the following section we describe the restructuring of the SPAM organization necessary in order to represent these kinds of knowledge.

2.2. Schematization of SPAM

In order to to make SPAM amenable to knowledge acquisition our approach has been to reduce the SPAM architecture to a set of generic control productions supported by scene-specific knowledge that can easily be generated by a program. Experimentation with the system architecture is now straightforward since the actual production generation is centralized in one program. Each piece of knowledge is encoded as a schema, with different schemata used to represent different types of knowledge. Schemas can easily be collected (or partitioned) to form new knowledge bases. Since the schemas are simply text files, it is trivial to combine different schemata to produce more complete knowledge bases. A discussion of this representation can be found in Section 4, a detailed description of the internals of schemata is found in

Appendix I, and examples of the generation of productions from a schema is illustrated in Appendix II.

This implementation has restructured SPAM such that within any interpretation phase, no rule has to know of the existence of any other rule. These intra-phase interactions were difficult to identify in the hand generated system, and made it very difficult to perform large wholesale changes to the knowledge base. Since there is a uniform interface to all rules within a particular phase, it is easier to allow users to specify interphase events such as calling consistency-checking within model-generation phase. The functional-area phase is an example of one part of the system that required some generalization for use on other domains. Originally developed with airports in mind, functional-areas had no shape constraints. However we have found cases in our suburban house scene experiment where shape constraints in addition to compatibility constraints are required. RULEGEN gives us the opportunity to easily propagate these changes to the different systems we have built. In the following section we briefly describe the ISCAN/RULEGEN/SPATS system organization.

3. Tools For Knowledge Acquisition In SPAM

There are several reasons why knowledge acquisition for SPAM appears to be relatively straight-forward. First, SPAM uses simple, pairwise tests to represent spatial consistency. Second, it is ordinarily easy for humans to characterize situations where special knowledge, either derived from further image analysis, or from additional consistency testing, can be used to disambiguate conflicting hypotheses. For example, if the two hypotheses differ in that one is above the ground plane, e.g.. a hangar or building, and the other is at the ground plane, a runway or road, then invoke an image analysis tool, stereo verification, to determine the preferred hypothesis.

The implications of the first observation is that the user is not forced by the architecture to conceptualize complex spatial consistency rules encompassing many primitives. For example, SPAM represents, *runways intersect taxiways that are oriented towards the tarmac* as two independent tests. This is partly to accommodate errorful image segmentation data which may not produce all of the primitives required for a more complex match, and a desire to not require complex matching of productions in our implementation language, OPS5[10, 11]. The second observation is a function of the task domain, the available image analysis tools, and our design of the SPAM architecture. A small set of several dozen geometric tests appear to suffice to represent the spatial relations that human users characterize as important for describing relationships between scene domain objects. Finally, we believe that it is possible to find images for a class of scenes, say, 20 commercial airports, which would allow us to acquire a cross-section of spatial organizations representative of commercial airports. This approach also lends itself toward exploring systems that would automatically synthesize interesting properties and learn the importance of various spatial relationships.

Knowledge acquisition systems range from interactive user dialogue via structure editors to acquisition systems that are tightly coupled with a task performance system. The degree to which the knowledge acquisition system itself utilizes knowledge may range from enforcing a particular knowledge representation, to a system which decides what to ask a user and asks for as little information as necessary to remedy specific problems[12, 13, 14]. ISCAN falls somewhere in this continuum, toward the former method. It primarily enforces a particular schema representation for various types of knowledge utilized by SPAM. However, it also uses knowledge of the SPAM architecture to recognize conflicts and missing or incomplete information. But it performs as an observer and does not elicit or suggest remedies. It is also decoupled from the performance system, partly do to the long execution times of SPAM[1], and partly due to what we believe is a complex task domain which makes credit assignment for particular actions of the system difficult to analyze.

In the remainder of this section we describe our first attempt at knowledge acquisition for aerial image interpretation using the ISCAN user interface, the RULEGEN compiler, and the SPATS performance analysis tool. Figure 3 shows a detailed organization of the knowledge acquisition system overview presented in Figure1.

3.1. The ISCAN User Interface

ISCAN currently supports two methods of knowledge acquisition. The first method is that of a structured editor which allows users to add, delete, or modify knowledge represented as schemata. The second method is the use of interactive image segmentation to generate metric information such as area, perimeter, distance, and shape descriptions. This information is then integrated into the schemata as values or ranges of values for various constraints. In either case the output of ISCAN session is a file containing the task specific schemata necessary to compile a SPAM system.

As a structured editor ISCAN allows the addition,

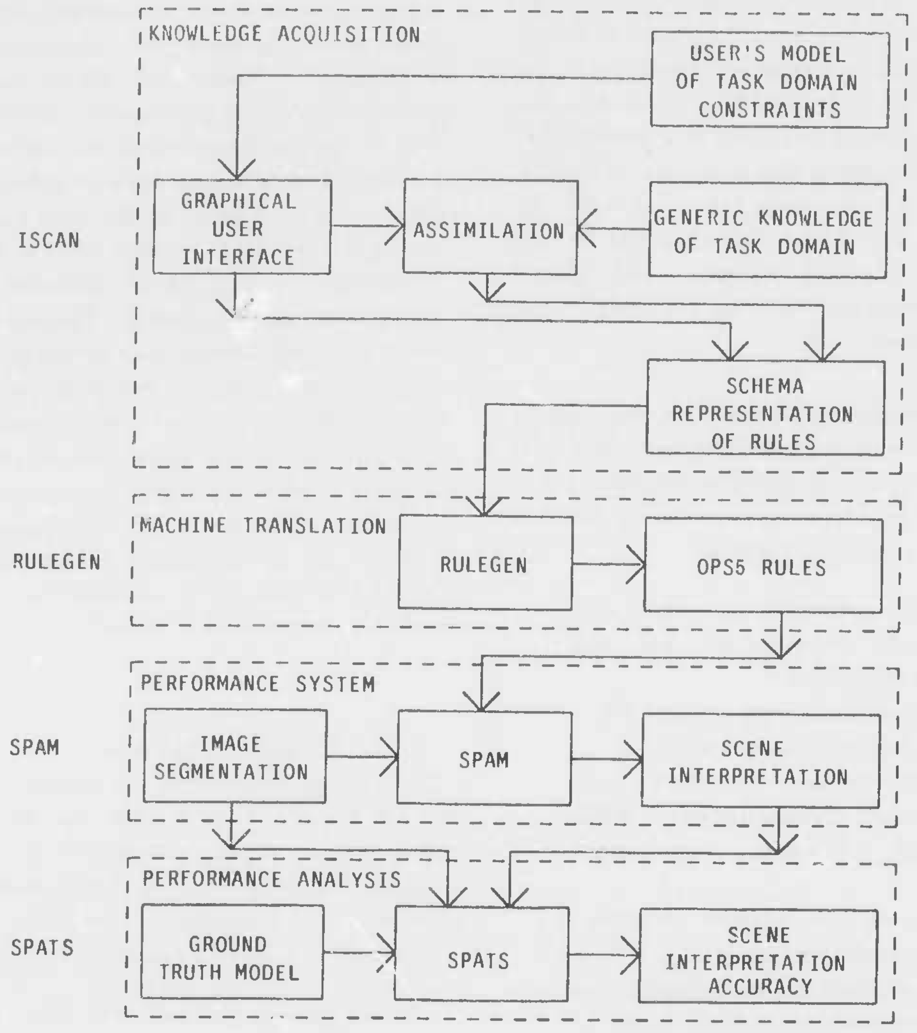


Figure 3-1: Knowledge Acquisition For SPAM

deletion, and modification of schemata for each phase in SPAM. It assists the novice user, asking questions to define the set of attributes for each schema and allowing example schemata to be displayed. ISCAN accommodates the more experienced user by foregoing the question/answer sessions and permitting the attributes to be entered directly. It maintains certain specific meta-knowledge such as knowing which attributes of a region must be computed and which can simply be matched. Much of the bookkeeping specific to the SPAM architecture is automated, thereby allowing the novice to concentrate on the task domain and not on whether attributes are filled in correctly. For example, some region attributes are precomputed, but some are too expensive to precompute for every region. It is really an implementation detail to know which attributes must be computed and which are precomputed.

Because the nature of the interpretation task is visual, ISCAN also provides a graphical interface for defining spatial relationship and performing measurements directly on an image. The user displays a representative image containing classes of objects or a particular site such as an airport. Associated with each image is a camera model[15, 16] that allows the graphics interface to generate constraints in terms of metric values rather than in image specific coordinates. For example a representative road width constraint can be specified as *between 10 and 15 meters* rather than *between 10 and 15 pixels*. The actual measurement is performed by ISCAN and is reported to the user. ISCAN can gather statistics over many examples to allow for a more robust range of constraints.

Since the measurements are always in terms of ground distances this allows for complete independence between the scale of the image under interpretation and the acquired knowledge base. This independence is a basic requirement for robust scene interpretation systems. With the scene constraints in mind, the user displays an image with characteristics of the general type of scene that is to be interpreted. After making measurements on the image directly, the user is questioned about the classes of objects in the scene, the shape characteristics of those objects, and their spatial relationships to one another. If this is a scene type that has previously been analyzed, it is possible that generic knowledge applicable to that scene can be applied. This can be added to the knowledge base being built. An example of this from the airport domain might be that every airport has at least one runway. This generic knowledge, if any, is coalesced with the knowledge about the scene given by the user.

ISCAN recognizes potential inconsistencies that may be generated during an interactive session or those that exist at the end of the session. It provides limited help in correcting such problems, a topic for future work. Some of kinds of inconsistencies recognized include:

- The inconsistent definition of class and subclass fragment interpretation and the violation of class hierarchies.
- The lack of a local-consistency schema for a class or subclass fragment interpretation.
- Multiple local-consistency schemata with identical fragment interpretations which potentially can generate inconsistent constraints.
- The omission of a subclass fragment interpretation from all functional areas.
- The definition of functional area descriptions and recognition of inconsistent combinations of component fragment interpretations.
- The specification of conflict resolution tests must be unique.
- The definition of model generation components that do not specify a method for selection.

We feel that the use of representative imagery to acquire general spatial relationships greatly increases a users ability to add such constraints to SPAM. This is primarily due to the ability to query ISCAN to display existing constraints involving fragment hypotheses and to detect conflicts or duplication than in a textual environment. Because the nature of our task is inherently visual, and visual tasks are done almost effortlessly by people, it appears that an it is easier for a person to give examples than to explain what is being extracted.

However, an area for future research for automating knowledge acquisition beyond user interaction is via learning by example. As we have discussed, our current work is focused on tools that aid in the translation of a users model of the task constraints into schemata. There are other sources of spatial knowledge that are amenable to automated extraction of constraints without user involvement. Figure 3-2 shows hand segmentations generated for use in performance analysis as *ground truth* data for Dulles International and Andrews AFB. Figure 3-3 illustrates a similar type of ground truth data, but perhaps not as detailed as the hand segmentations. It is generally available to aircraft pilots as Flight Information Publications, or FLIPcharts[5], published by the FAA and the Defense Mapping Agency. The goal of such research is to uncover spatial constraints by examining a large number of examples of airports whose spatial relationships are made explicit in either of these formats. Such a system must not only develop reasonable ranges of values such as *'airports have at least one runway, but no more than 7'* but must develop the subset of useful spatial relationships from the set of all possible relationships. We plan to explore this approach as a method to expand the scope of knowledge acquisition in ISCAN.

3.2. The RULEGEN Compiler

With the scene knowledge encoded as schemata, we need to put it into a form that can be utilized by our interpretation system. RULEGEN is a compiler which performs schema-to-production translation. This compiler has procedural knowledge of the SPAM control structure. Some of the functions the compiler must perform include:

- Efficiently initializing each rule so that large conflict-sets do not slow down the OPS5 conflict resolution process.
- The automatic generation of error-checking productions to make the system more robust, and trace productions for performance analysis.
- Managing control productions to efficiently match all the desired data in working-memory.
- The generation of interface functions for computations such as image analysis and geometric computation performed outside of OPS5 environment.
- The generation of data-structures representing the boundary values of the scene constraints.

In the SPAM architecture, region interpretations come mostly bottom-up, with top-down prediction and verification. With processing going in two directions, the management of control in a production system is non-trivial. RULEGEN handles the rule interactions and the order of rule firings by generating appropriate control rules to achieve the desired results. The data-structures and control productions vary from phase-to-phase because the

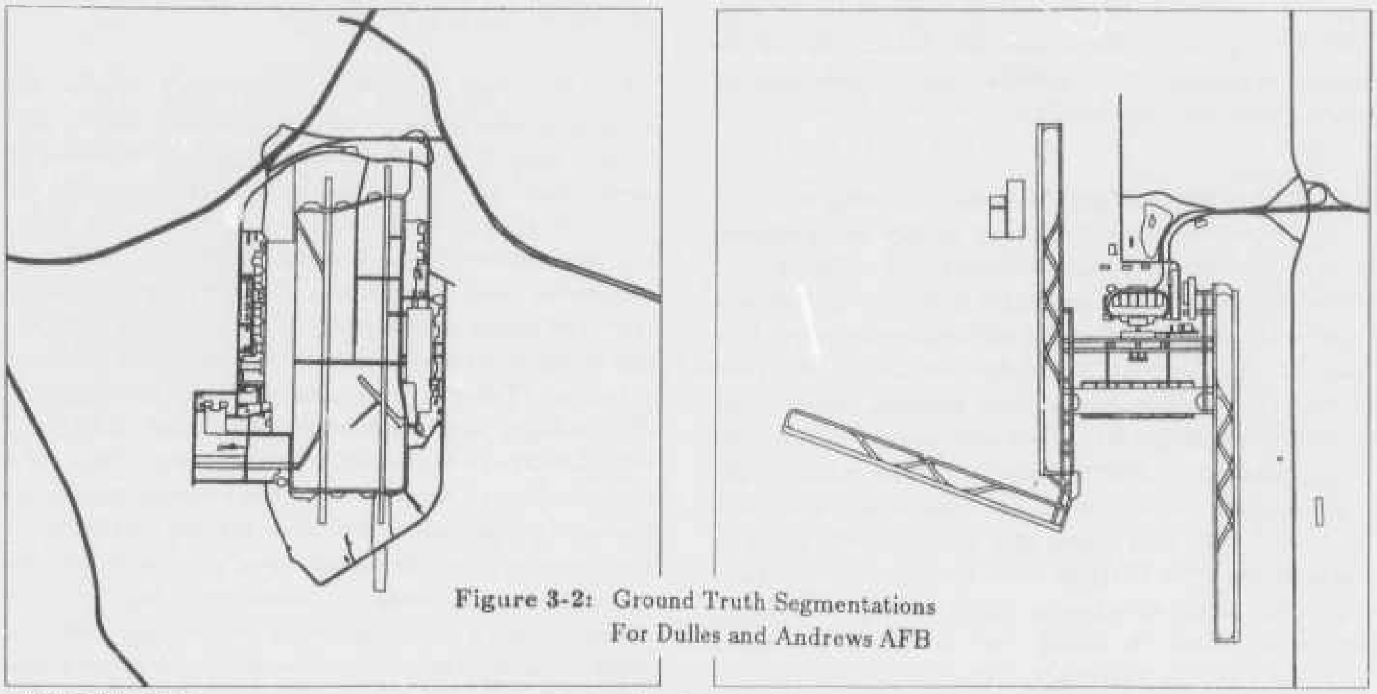


Figure 3-2: Ground Truth Segmentations For Dulles and Andrews AFB

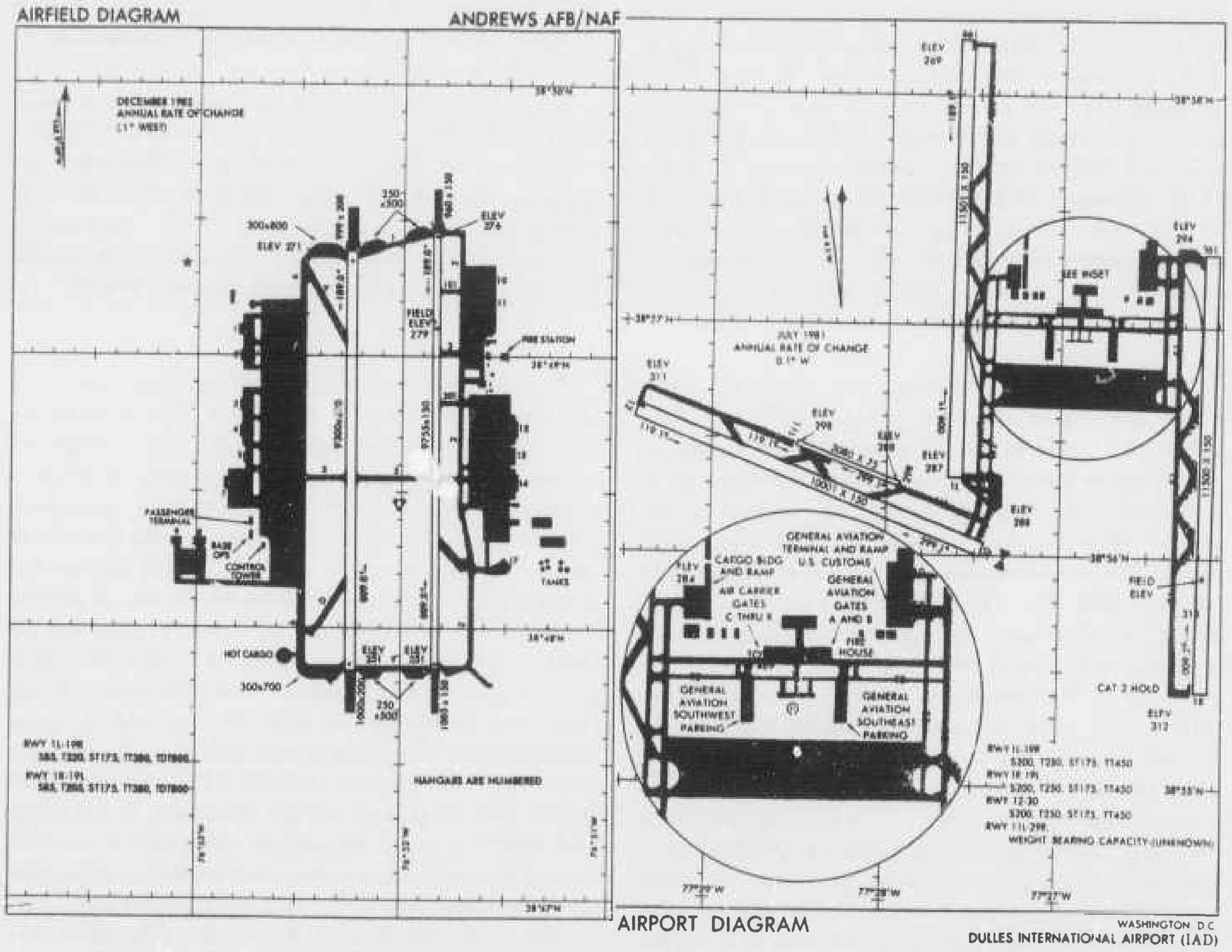

AIRFIELD DIAGRAM
ANDREWS AFB/NAF
DECEMBER 1982 ANNUAL RATE OF CHANGE (1° WEST)
PASSENGER TERMINAL
BASE OPS
CONTROL TOWER
HOT CARGO
FIRE STATION
TANKS
HANGARS ARE NUMBERED
FIELD ELEV 279
AIRPORT DIAGRAM
WASHINGTON D.C.
DULLES INTERNATIONAL AIRPORT (IAD)
JULY 1981 ANNUAL RATE OF CHANGE 0.1° W
SEE INSET
GENERAL AVIATION TERMINAL AND RAMP U.S. CUSTOMS
CARGO BLDG AND RAMP
AIR CARRIER GATES C THRU K
GENERAL AVIATION GATES A AND B
FIRE HOUSE
GENERAL AVIATION SOUTHWEST PARKING
GENERAL AVIATION SOUTHEAST PARKING
CAT 2 HOLD
FIELD ELEV
WEIGHT BEARING CAPACITY (UNKNOWN)

Figure 3-3: Flight Information Charts With Airport Layouts For Dulles and Andrews AFB

type of processing that occurs in each phase is very different. Appendix II gives some detailed examples of the actual expansion of a schema into a collection of productions executable by SPAM.

3.3. SPATS: Automating Performance Analysis

An often overlooked component of any interpretation system are tools that aid in incremental refinement of knowledge and in the measurement of the effects of various types of knowledge within the performance system. As a part of our work in knowledge acquisition we have developed a performance analysis program, SPATS, that gives us some insight into the overall accuracy of the scene interpretation. SPATS uses a region-based hand segmentation with correct interpretation attributes associated with each region as a baseline with which to compare the SPAM interpretation. In the case of machine-segmented data, we compute region overlap with hand-segmented data in order to generate a correct interpretation. In some cases, ambiguous results must be resolved manually before statistics can be generated. A log file generated by SPAM at each phase of interpretation is used to acquire the internal state of the SPAM interpretation. At a gross level we need a consistent method to measure the accuracy of scene interpretations generated with alternative or refined knowledge. In terms of the 'debugging' of knowledge, we require an indication of where one might spend time improving the knowledge base to improve scene interpretation. SPATS attempts to summarize the important performance statistics in a succinct manner for each phase of processing by SPAM.

For the *region-to-fragment* and *local-consistency* phases, we require statistical measures that accurately reflect the performance of geometric knowledge in classifying the initial image segmentation. Factors such as the number of competing hypotheses, as well as the number of correct and incorrect hypotheses, are most useful. SPATS provides this information in tabular form (see Appendix III). This information can be compared within, as well as across class and subclass boundaries, to give some indication of the effectiveness of the geometric constraints. One measure, the correct branching factor, defines how many interpretations there were, on the average, for each correct interpretation. This branching factor increases from zero (with zero signifying no correct interpretations) as the number of competing hypotheses increases. As this number increases, the effectiveness of the associated geometric knowledge decreases. To rectify this problem, we would then try to isolate which class or subclass constraints were too weak and attempt to tighten them, or look for new sources of knowledge that would increase our ability to discriminate.

For the *functional-area* phase, SPATS checks the integrity of the functional-areas generated by SPAM. This involves using the functional area declarative knowledge to check that all the fragment interpretations fit the definition of that functional-area type. Statistics giving the correctness of each functional-area, the number of compatible and incompatible fragment interpretations contained within the boundary of the functional area, but not found to be components of the functional area, are generated. This gives us a measure of the cohesiveness of the functional area. One would expect small numbers of incompatibles to be present, with some compatibles. If a large number of compatible fragment interpretations are present questions about why they did not participate as members of the functional area can lead to the modification of geometric consistency rules or the recognition that knowledge of new relationships should be sought. Finally, these mismatches between functional area definitions and geometric consistency can indicate whether the user's definition of a functional-area is appropriate.

For *model-generation* phase, the constituent functional-areas of the various scene models are compared. This is done as a first attempt at quantifying the differences between each of the scene models, if more that one consistent model is generated by SPAM. Currently a complete analysis involving the accuracies of the included interpretations has not yet been completed in SPATS.

As a general methodology for the evaluation of a knowledge base we have found that running SPAM on hand-segmented ground truth region data is a valuable test of the interpretation process. It presents to SPAM a "perfect" low-level segmentation, effectively decoupling the low-level image analysis. The results from this type of experiment can be used to argue the issue of whether the interpretation problem is fundamentally one of dealing with errorful segmentations and should be remedied by working to improve the segmentation. We believe that even a 'good' low-level segmentation requires significant high-level knowledge in order to generate a scene interpretation. The use of ground truth data also makes it easier to avoid a common problem exhibited by computer vision systems of unknowingly developing intermediate and high-level vision components which rely on machine segmentations that can be characterized as over-segmented or under-segmented. In our view, one should at least make explicit these assumptions if they are a factor in the interpretation process.

SPATS is a useful tool for indicating gaps, weaknesses, or inconsistencies in various types of knowledge in SPAM. We believe that a statistical approach must be used due to the large number of segmentations involved in the interpretation process, as well as the inaccuracies in assigning interpretations to those segmentations. Future research is being focused in the refinement and addition of new measures, particularly in *model-generation* phase, and looking at techniques for making performance analysis a more active component of SPAM.

In the following Section we give some detailed examples of the schema-based knowledge representation generated by ISCAN and used by RULEGEN to compile SPAM systems. Section 5 describes the use of ISCAN/RULEGEN to generate a SPAM system for a new suburban house scene task.

4. A Schema-Based Knowledge Representation

Our schema-based knowledge representation is one method for linking knowledge acquisition as performed in ISCAN with knowledge utilization in SPAM. The focus of this work was to develop an intermediate representation for the domain specific knowledge used by SPAM as a target description for knowledge acquisition and as a source description for automatic generation of the interpretation system. Therefore, some important properties for the representation are as follows:

- Sufficiently general to represent the kinds of knowledge and spatial relationships utilized in each of the SPAM interpretation phases.
- Could be compiled into our target production system language, OPS5.
- Easily organized or partitioned into independent knowledge sets.
- The format is understandable by non-programmers. The knowledge and its purpose should not be obscured by the implementation language.

The schema-based representation has been conducive to experimentation. It is far easier to add and delete knowledge and to measure the effect on system performance, as the rule generation is performed automatically; only the control productions that embody the SPAM architecture are hand crafted. Improvements in structuring rules are easy to propagate. If an improved method is developed, the generating functions are modified appropriately and all generated productions are updated. Since our goal is to produce a working system that handles a variety of aerial interpretation tasks it is easier to test the generality of the SPAM architecture if we can generate task specific systems. Hand generation of SPAM is not an attractive alternative, especially in light of our requirement to perform experimentation by adding and removing specific types of domain knowledge.

4.1. A House Fragment Rule

The following is an excerpt from the RULEGEN schema file for the *region-to-fragment* phase of SPAM, used to interpret suburban-housing scenes. This set of attribute-value pairs will generate data-structures and productions allowing fragment interpretations for regions of type 'house' to occur.

```
'CLASS'                = 'house'
'REGION-DEPENDENCES'   = ''
'FRAG-DEPENDENCES'     = 'object-type compact
                          && hypothesis unknown'
'SHAPE-CONSTRAINT'     = 'area
                          && 50.00 <= value <= 150.00'
'SHAPE-CONSTRAINT'     = 'ellipse-length
                          && 12.00 <= value <= 18.00'
'SHAPE-CONSTRAINT'     = 'ellipse-width
                          && 10.00 <= value <= 20.00'
'SHAPE-CONSTRAINT'     = 'ellipse-linearity
                          && 0.00 <= value <= 3.50'
```

Each of the schema attributes are described below.

CLASS
: Determines the class of object to which this set of constraints is applicable. In this case, the rule defined will apply to houses.

REGION-DEPENDENCES
: Makes sure that a given set of attributes have been computed *before* any house interpretation rules can be fired. In this case, there are no computed attributes that must exist before this rule can fire.

FRAG-DEPENDENCES
: Allows a constraint rule to depend on the success of a previous constraint rule. In this case, the constraint rule for the object type "compact" must have previously executed successfully (e.g. a compact interpretation created) in order for this rule to fire.

SHAPE-CONSTRAINT
: Defines the actual constraints that comprise the class definition. Any number of these constraints can occur here. Currently, 2-D shape characteristics, intensity characteristics, depth measures, and texture measures fall into this category. In this case, four constraints completely define the class-type house. For example, the first constraint limits houses to have areas between approximately 50 and 150 square meters.

The first three attributes generate a small set of control productions that determine if this rule applies to a given region. Each SHAPE-CONSTRAINT generates a single production for the particular constraint given.

When this rule becomes applicable, an initialization production fires and creates a subtask which is matched by each of the constraint productions. All of these productions are allowed to fire, each determining the "goodness" of the match for a single constraint. Finally, a domain-independent production looks at the accumulated scores and decides whether a house interpretation should be made. Appendix I gives a detailed description of the each schema-type and spatial constraint currently available in ISCAN. Appendix II contains the actual productions generated by RULEGEN for each of the schemata in Section 4.1 and Section 4.2.

4.2. A House-Road Consistency Rule

For the second phase of SPAM, local-consistency-check, RULEGEN uses a new set of attributes to define a rule. The basic idea, as discussed in Section 2, is to generate pairwise tests that exploit the fact that although there may be a large number of errorful fragment hypotheses, only small numbers will be mutually consistent. The following is one such l consistency test, *houses-are-parallel-to-roads*, others might include, proximity of houses to each other, distance from roads, orientation of a house to a driveway, etc.

```
'RULENAME'    = 'houses-are-parallel-to-roads'
'CONFIDENCE' = '0.8'
'HYPOTHESES' = 'house && road'
'GEOMETRICS' = 'orientation'
'SUBTYPES'   = 'parallel'
'BOUNDS'     = '0.00 <= value <= 0.50'
```

RULENAME
: Attaches a unique, human-readable name to the rule, so we know what it does. In this case, this rule will determine whether a road is parallel to a house.

CONFIDENCE
: Assigns a confidence value to this rule, which (subjectively) describes its discrimination ability. It is a number between 0 and 1, with a value of 1 implying that the rule can perfectly (uniquely) determine that the participating interpretations are correct. In this case, the rule is believed to be a reasonably good characterization of a house and a road. It is given a (subjective) confidence value of 0.8.

HYPOTHESES
: Defines the classes of interpretations to which this rule applies. In this case, the rule applies to the relationship between houses and roads.

GEOMETRICS
: Translates the high-level spatial relation into a low-level geometric test. In this case, "parallel" is translated as "orientation".

SUBTYPES
: Further defines the translation of the high-level spatial relation. In this case, "parallel" implies that the orientation of the interpretations is being tested (see the previous description), and that the particular type of orientation test should be "parallel".

BOUNDS
: Completes the definition of the high-level spatial relation by defining the error tolerance. In this case, the geometric test has a tolerance of 0.5 radians.

The first three attributes define the high-level significance of this rule. The last three attributes describe, in low-level terms, the high-level intentions. For this example, the rule may be read as *houses being parallel to roads* means for a particular house, a road must be oriented parallel to that house, within a tolerance of 0.5 radians. Thus, a house fragment hypothesis and a road fragment hypothesis will support each other if this test is successful.

4.3. A House Functional-Area Definition Rule

The functional-area phase of SPAM groups individual hypotheses into mutually supporting collections of hypotheses that represent meaningful sub-parts of the overall scene model. The knowledge in this phase defines these scene sub-parts.

```
    'FA-NAME'        = 'house-area'
    'SEED-REGION'    = 'house'
    'DEFINITION'     = 'driveway && grassy-area'
```

FA-NAME
: Assigns a name to this functional-area definition.

SEED-REGION
: Defines the principle hypothesis of a functional-area. If this type of hypothesis does not exist, no functional-area of this type can exist. Here, we designate the interpretation type 'house' as our seed region.

DEFINITION
: Enumerates the possible constituents of this type of functional-area. The functional-area 'house-area' can contain only houses, driveways, and grassy-areas.

4.4. A Suburban-Scene Model Rule

For the final phase of processing, model-generation, SPAM combines functional-areas together to form models of the entire scene. During this process, conflicting interpretations will be identified and must be resolved in order to obtain a final, consistent model. Knowledge about the types of conflicts, and ways to disambiguate them, is encoded in this phase.

```
    'CONFLICT'       = 'house && driveway'
    'RESOLUTION'     = 'function && stereo'
```

CONFLICT

This attribute specifies the name of the conflict type that needs special attention in order to be disambiguated. In the example at hand, house-driveway conflicts will be specifically addressed.

RESOLUTION

Defines the type of process to use to do the disambiguation. In this case, we know that houses have height and driveways do not, therefore invoke a stereo process to determine whether or not the region has height.

Those conflicts not enumerated are handled by a default resolution strategy that takes into account the confidences of the individual interpretations, as well as the amount of support for each interpretation in the context of the current scene model.

5. A New Task Domain For SPAM

In this section we will give a brief example of one of the interpretation tasks used in our experiments to date. We show some results of the interpretation of a suburban house scene by SPAM built completely using the ISCAN/RULEGEN system. Figure 5-1 is a photograph of three of the suburban house scene images used by Hwang[17] at University of Maryland. Our intent was to replicate this work using the ISCAN/RULEGEN system to generate a SPAM with spatial knowledge of suburban house scenes. Figures 5-2 and 5-3 show a human segmentation and machine segmentation of one of the six suburban house scenes used in this experiment. This replicates work performed by Hwang[17] on this image set. The goal is to segment and identify the houses, roads, grassy areas, and driveways in the aerial image. This is a somewhat simpler task that the original airport scene interpretation task performed by SPAM, but it turned out to be of reasonable scale to evaluate and refine ISCAN/RULEGEN.

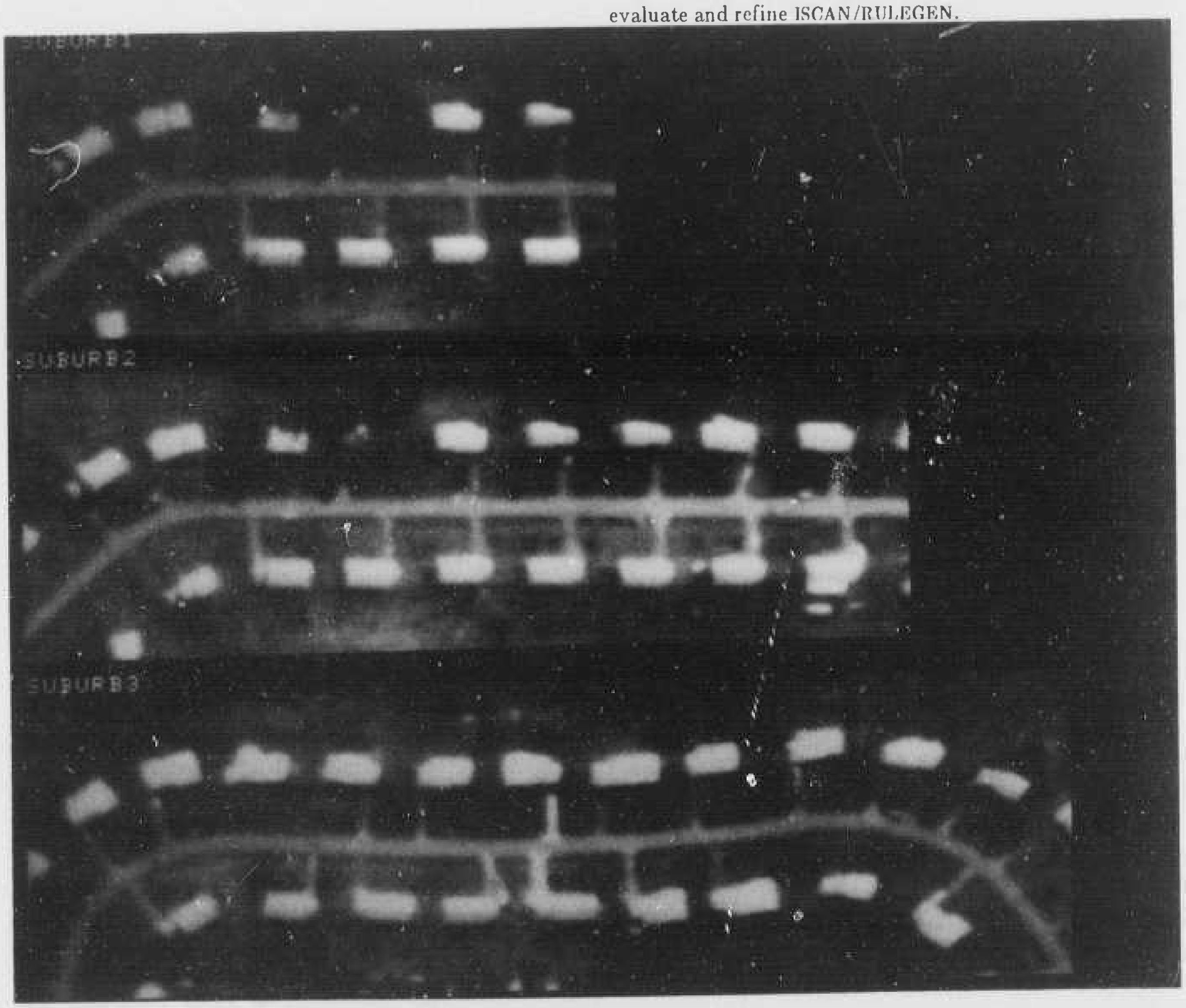


Figure 5-1: Suburban House Scene Imagery

The SPAM knowledge base for these images were developed with measurements using ISCAN on two training images from the set. The knowledge base was refined iteratively, first using the hand segmented set of regions, then modified using machine segmentation data. Currently, SPAM has been run on using both hand and machine segmentations for three of the six images, the two training and one test image. The image in this example is the test image.

The purpose of the hand segmentation is to provide "ground truth" for our automated analysis programs that are used to generate region-by-region interpretation statistics used to measure SPAM's performance in region labeling and overall scene interpretation. We have also found it useful to run the hand segmentations through SPAM in early phases of rule development in order to completely decouple the low-level image analysis from the interpretation system. This noise-free approach allows us to uncover gross omissions or unexpected interactions between the local consistency rules.

Figure 5-3 is the result of running our image segmentation system, MACHINESEG[18], which uses region-growing and shape extraction simultaneously to look for characteristic linear, compact, and blob regions. Although the image is relatively uncomplicated several houses are missed, some are only partially segmented, and the roads and driveways are oversegmented into multiple pieces. However, this is reasonable in the context of current computer vision segmentation capability. Figures 5-4 and 5-5 show functional areas generated by SPAM for houses and roads, respectively. Figure 5-4 shows the functional area generated from the hand segmentation in Figure 5-2 including regions whose fragment interpretation were 'house' or 'grassy area' Figure 5-5 shows the functional area including 'roads' and 'driveway' hypotheses for the machine segmentation in Figure 5-3. We feel that the functional areas are quite good good in both cases and are similar to results generated by Hwang[17]. While direct comparisons of two knowledge-based systems using different methodologies are not the subject of this paper, it is important to point out that these results were generated by automatic compilation of user-defined knowledge tailored to the suburb house scene task within the framework of the SPAM interpretation architecture.

5.1. Structural Differences In Hand versus Machine Generation

One goal for RULEGEN was to be able to reproduce the hand written version of SPAM reported on in[1, 2] for airport scenes. This system, which we will call SPAM-1, contained over 500 hand-coded OPS5 productions, and was used to interpret airport scenes of National Airport, Los Angeles International and NASA AMES Moffett Field. In contrast SPAM-2 was built with the RULEGEN compiler by manually extracting the primitives and constraints from SPAM-1 and encoding them as schemata. The ISCAN system was not used to build SPAM-2 since the knowledge was readily available in the existing SPAM-1 productions. However, some of the experience gained in building SPAM-2 was used in the construction of the ISCAN system. SPAM-2 was verified on the same airports as SPAM-1 giving quite similar results. It has since been used on other airports, not tested with SPAM-1, such as Dulles International, Andrews Air Force Base, and San Francisco, with mixed results. SPAM-2 is now the basis for future work in airport scene analysis. SPAM-3 is the suburban house scene system and was built entirely using ISCAN and RULEGEN.

Figure 5-6 gives a breakdown of productions comprising each of the SPAM interpretation phases for each of the three systems. With this data, we will try to characterize some of the differences between SPAM-1 and SPAM-2 and characterize the emergence of domain independent knowledge as a result of the restructuring of the SPAM architecture as described in Sections 2.2 and 4. To do a proper comparison of SPAM-1 to SPAM-2, one must add the number of domain-independent productions to the number of generated productions for SPAM-2. For example, if we do the comparison for the *region-to-fragment* phase (RTF), we find that there are 120 productions in the hand-coded system and 91 productions in the machine-coded system.

The decrease in the number of productions is somewhat due to the experience gained in during the hand-coding of SPAM-1[11] applied to RULEGEN. In addition, the desire to generalize the SPAM architecture forced us to consider how to gain efficiency as well as generality. The decoupling of domain-dependent knowledge from the SPAM control rules actually lead to a decrease in the number of OPS5 productions being generated. In the case of the last two phases, *functional-area* and *model-generation*, it is clear that most of the knowledge is now encoded by the domain-independent rules or migrated to procedural knowledge. This is due to the more abstract functions provided by these phases, such as grouping, merging, and splitting which appear to be task independent and are not knowledge intensive. Thus the bulk of the domain knowledge appears to be in the *region-to-fragment* and *local consistency* phases. Once fragment interpretations are generated along with associated chains of consistent relationships the aggregation of these fragments into functional areas is now mostly procedural. What

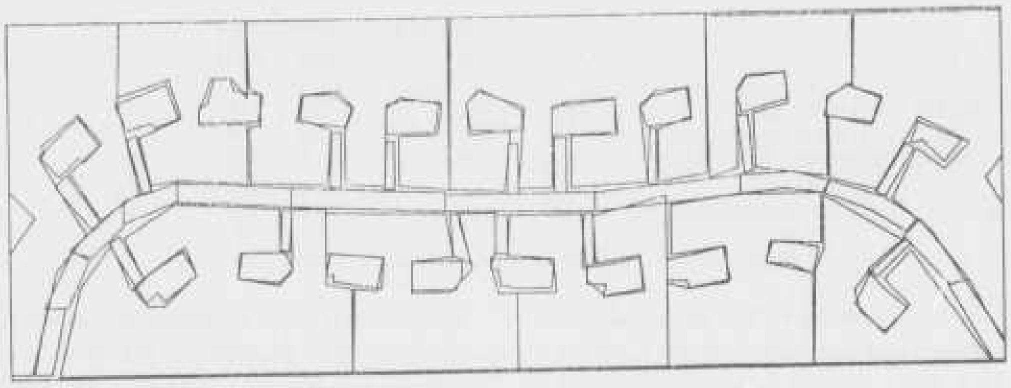


Figure 5-2: A Hand segmentation of a suburban scene

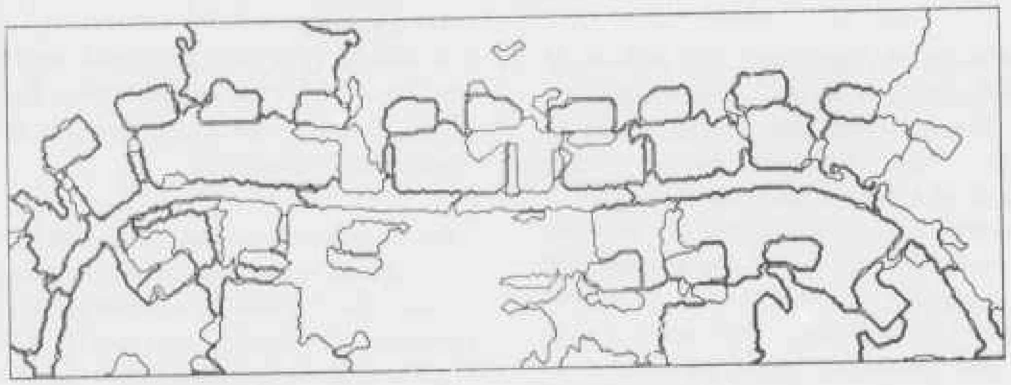


Figure 5-3: A Machine segmentation of a suburban scene

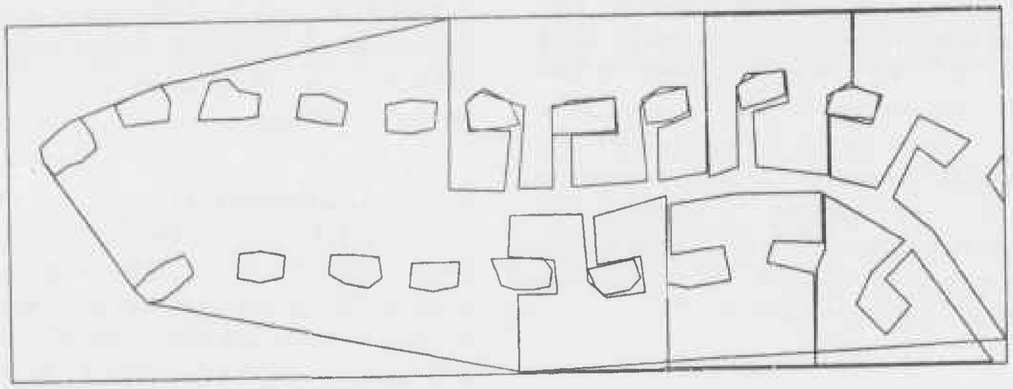


Figure 5-4: A House functional-area result from hand segmentation

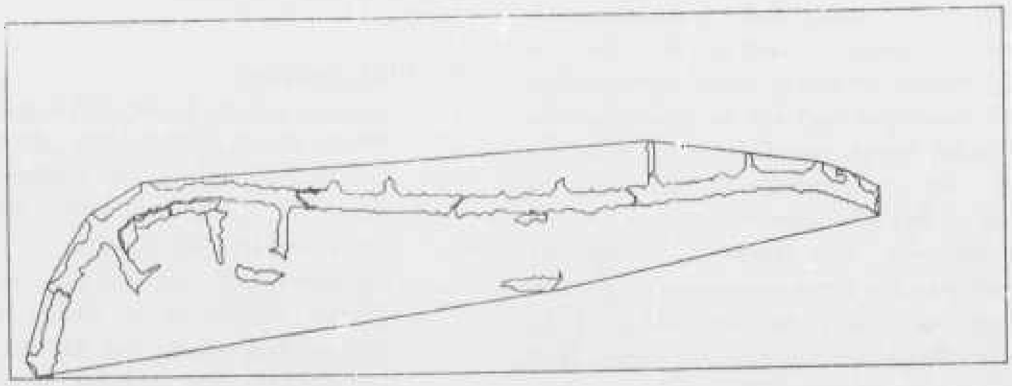


Figure 5-5: A Road functional-area result from machine segmentation

Task	Interpretation Phases RTF	LCC	FA	MG	MISC
(SPAM-1) Airport (Hand)	120	304	23	50	16
(SPAM-2) Airport (Rulegen)	54	297	1	6	0
(SPAM-3) Suburb (Rulegen)	32	99	1	1	0
Rulegen Domain Independent	37	7	11	35	7

Figure 5-6: Rules generated by Interpretation Phase

knowledge remains is the definition of the functional area groups, model definitions, and methods to resolve conflicts. The net result is a more general system with fewer productions and, though not explicit from this data, faster execution times.

For the suburban-house scene task, the amount of knowledge required appears to be significantly less than for the airport task. This is not surprising since the number of productions in the *region-to-fragment* and *local-consistency* phase is directly related to the number of geometric and spatial constraints used to interpret the scene. A simpler scene type intuitively implies that fewer scene primitives are present and that a smaller number of spatial constraints are available. One would expect, therefore, that the amount of knowledge required to interpret the less complex scenes would decrease from that required for the more complex ones. This is exactly the case for comparisons of SPAM-2 and SPAM-3. However, even in the case of the suburban house task the pattern of the preponderance of knowledge as reflected in productions appears in the first two interpretation phases as seen in for the airport task. A more precise characterization of the amount of knowledge needed to interpret a particular scene type, related to a measure of apparent task complexity, would be an interesting result from this work. This may become possible as more task domains are implemented within the SPAM architecture.

6. Conclusion

In this paper, we have described a collection of tools for knowledge acquisition, automated compilation of knowledge, and performance analysis for SPAM, a knowledge-based system for aerial image interpretation. Several types of knowledge that can be expected to be important for aerial image interpretation systems are described. The use of knowledge in SPAM and its representation as schemata for knowledge acquisition and compilation is discussed. The results of a completely automated generation of a SPAM system for a new task domain are described. Some preliminary analysis of the effects of decoupling domain-independent knowledge from the interpretation system are presented,

In summary, by focusing on automated knowledge acquisition and compilation we have generated a more manageable interpretation system for experimentation and measurement. This flexibility gives us the capability to investigate the automated construction of knowledge-based image interpretation systems for a variety of tasks. It is difficult to envision how SPAM could have progressed from its initial 'hand coded' version to a more general system capable of performing multiple tasks without the development of these tools.

Future research includes expanding the range of aerial image interpretation tasks performed using the new SPAM architecture. We are also interested in the development of techniques for further automation of the knowledge acquisition process by using collections of hand-segmented imagery and existing large scale databases such as the FLIPcharts described in Section 3.1. One goal is to investigate the use of more knowledge intensive techniques for knowledge acquisition toward systems capable of automatic selection of scene primitives and important spatial relationships.

7. Acknowledgments

Larry Eshelman and John McDermott provided some tough comments on an early version of this paper. Lambert Wixson and Brian Yamauchi implemented major portions of SPATS and ISCAN. Robert Lai aided with the preparation of this paper. Thanks to Bob Simpson for inviting us to present it at the DARPA Image Understanding Workshop.

8. Bibliography

1. McKeown, D.M., Harvey, W.A. and McDermott, J., "Rule Based Interpretation of Aerial Imagery," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. PAMI-7, No. 5, September 1985, pp. 570-585.
2. McKeown, D.M., McVay, C.A., and Lucas, B. D., "Stereo Verification In Aerial Image Analysis," *Optical Engineering,* Vol. 25, No. 3, March 1986, pp. 333-346, Also available as Technical Report CMU-CS-85-139

3. Froesch, C., and Prokosch, W., *Airport Planning,* John Wiley and Sons, Inc., New York, N. Y., 1946.

4. Horonjeff, R., and McKelvey, F. X., *Planning and Design of Airports,* McGraw-Hill Book Company, New York, N. Y., 1983, Third Edition

5. Defense Mapping Agency, "Flight Information Publication: Instrument Approach Procedures and Airport Diagrams," St. Louis Air Force Station, MI 63118, 7 July 1983.

6. H.G. Barrow and H. Popplestone, "Relational Descriptions in Picture Processing," *Machine Intelligence,* Vol. 6, 1971, pp. 377-396.

7. J.M. Tenenbaum and S. Weyl, "A Region-Analysis Subsystem for Interactive Scene Analysis," *International Joint Conference on Artificial Intelligence,* Vol. 2, 1975, pp. 682-687.

8. A. R. Hanson and E. M. Riseman, *VISIONS: A computer system for interpreting scenes,* Academic Press, New York, 1978, pp. 303-333.

9. Robert Belknap, Edward Riseman, and Allen Hanson, "The Information Fusion Problem and Rule-Based Hypotheses Applied to Complex Aggregations of Image Events," *DARPA Image Understanding Proceedings,* 1985, pp. 279-292.

10. Forgy, C. L., "The OPS5 User's Manual," Tech. report, Carnegie-Mellon University, Department of Computer Science, 1981.

11. Brownston, L., Farrell, R., Kant, E., and Martin, N., *Programming Expert Systems in OPS5,* Addison-Wesley, Reading, MA, 1985.

12. Buchanan, B.G., "Some Approaches to Knowledge Acquisition," Tech. report STAN-CS-85-1076, Stanford University, July 1985.

13. Larry Eshelman and John McDermott, "MOLE: A Knowledge Acquisition Tool That Uses Its Head," *Proceedings of the National Conference on Artificial Intelligence,* 1986, .

14. John McDermott, "Making Expert Systems Explicit," *Information Processing 86,* H.J. Kugler, ed., Elsevier Science Publishers B.V., 1986, pp. 539-544.

15. McKeown, D.M., "MAPS: The Organization of a Spatial Database System Using Imagery, Terrain, and Map Data," *Proceedings: DARPA Image Understanding Workshop,* June 1983, pp. 105-127, Also available as Technical Report CMU-CS-83-136

16. McKeown, D.M.,, "Digital Cartography and Photo Interpretation from a Database Viewpoint," in *New Applications of Databases,* Gargarin, G. and Golembe, E., ed., Academic Press, New York, N. Y., 1984, pp. 19-42.

17. Shang-Shouq Vincent Hwang, *Evidence Accumulation for Spatial Reasoning in Aerial Image Understanding,* PhD dissertation, University of Maryland, 1984.

18. McKeown, D.M., Denlinger, J.L., "Map-Guided Feature Extraction from Aerial Imagery," *Proceedings of Second IEEE Computer Society Workshop on Computer Vision: Representation and Control,* May 1984, Also available as Technical Report CMU-CS-84-117

Appendix I

A short description of the attributes available for each phase, and their legal values, follows. The available geometric and spatial relationships are also given.

Region-to-Fragment

For the region-to-fragment phase, knowledge about the expected shape of the classes of objects appearing in the scene is encoded. The `<region-attribute>` is a characteristic computed for each of the segmentation regions coming from the segmentation process, whether hand or machine.

```
'CLASS'                = '<hypothesis>'
'REGION-DEPENDENCES'   = '<any string>'
'FRAG-DEPENDENCES'     = '<any string>'
'SHAPE-CONSTRAINT'     = '<region-attribute>
                          && <range>'
           <any number of shape-constraints>
```

The following is a sample region-to-fragment schema used by the suburban-house scene version of SPAM. RULEGEN uses this schema to produce productions that define, via shape-characteristics, the subclass "house" within the SPAM system.

```
'CLASS'                = 'house'
'REGION-DEPENDENCES'   = ''
'FRAG-DEPENDENCES'     = 'object-type compact
                          && hypothesis unknown'
'SHAPE-CONSTRAINT'     = 'area
                          && 50.00 <= value <= 150.00'
'SHAPE-CONSTRAINT'     = 'ellipse-length
                          && 12.00 <= value <= 18.00'
'SHAPE-CONSTRAINT'     = 'ellipse-width
                          && 10.00 <= value <= 20.00'
'SHAPE-CONSTRAINT'     = 'ellipse-linearity
                          && 0.00 <= value <= 3.50'
```

The attributes available to characterize the geometric constraints for a single scene primitive are summarized below. Most of these attributes are precomputed prior to being loaded into the interpretation system. The others are computed as they are needed. For example, if texture measures are used only to discriminate between the different subclasses of the class called blob, then texture need only be computed for that much smaller subset of regions that are interpreted as blob regions.

<region-attribute>	<range>	<status>
texture-low	[0 - 100]	dynamically-computed
texture-moderate	[0 - 100]	dynamically-computed
texture-high	[0 - 100]	dynamically-computed
location-lat	[0 - 10000000]	precomputed
location-lon	[0 - 10000000]	precomputed
orientation	[0 - 2pi]	precomputed
ellipse-width	[0 - 5000]	precomputed
ellipse-length	[0 - 10000]	precomputed
mbr-width	[0 - 5000]	precomputed
mbr-length	[0 - 10000]	precomputed
depth-low	[0 - 100]	dynamically-computed
depth-moderate	[0 - 100]	dynamically-computed
depth-high	[0 - 100]	dynamically-computed
curvature	[0 - 1]	dynamically-computed
ellipse-linearity	[0 - 1000]	precomputed
mbr-linearity	[0 - 1000]	precomputed
compactness	[0 - 1]	precomputed
fractional-fill	[0 - 1]	precomputed
area	[0 - 10000000000]	precomputed
perimeter	[0 - 10000000]	precomputed

Local-Consistency

We now describe the attributes and geometric relations used in defining a local-consistency rule. The knowledge represented makes explicit ambiguous spatial/relational concepts such as "close-to", "oriented-toward", or "far-from". This is done by imposing bounds on each spatial relation, and using a confidence function to smooth out the discontinuities associated with simply using thresholds.

```
'RULENAME'    = '<any string>'
'CONFIDENCE'  = '[0 - 1]'
'HYPOTHESES'  = '<hypothesis1> && <hypothesis2> && ..'
'GEOMETRICS'  = '<spatial-relation>'
'SUBTYPES'    = '<sub-relation>'
'BOUNDS'      = '<range>'
```

An example local-consistency schema follows, which defines the rule that houses should be parallel to roads.

```
'RULENAME'   = 'houses-are-parallel-to-roads'
'CONFIDENCE' = '0.8'
'HYPOTHESES' = 'house && road'
'GEOMETRICS' = 'orientation'
'SUBTYPES'   = 'parallel'
'BOUNDS'     = '0.00 <= value <= 0.50'
```

The set of possible primitive spatial relations are listed below. This small set has been found to be expressive enough to describe local-consistency relations for the scenes SPAM has interpreted thus far i.e. the airport and suburban-housing scenes.

<spatial-relations>	<sub-relations>	<range>
distance	centroid	[0 - 10000]
	average	"
	least	"
	greatest	"
orientation	toward	[0 - pi]
	parallel	"
	perpendicular	"
intersection	nil	[t, nil]
overlap	nil	[0 - 1]

<keywords>	<keyword-data>
default	none
function	name of function used to do resolution
conclusion	name of function used to combine results

Functional-Area

The knowledge encoded in the functional-area phase is somewhat implicit. It is represented by the associations made between hypotheses when one defines a functional-area type. The associated objects are located in close, physical proximity to one another and have similar functions. Each of the FA-NAME attributes defines a functional-area type which will be used as a part of the overall scene model.

```
'FA-NAME'     = '<any string>'
'SEED-REGION' = '<hypothesis>'
'DEFINITION'  = '<hypothesis1>
                && <hypothesis2> && ...'
```

The following functional-area schema defines the functional-area type *terminal* as being composed of terminal-building, road, parking-lot, and parking-apron hypotheses.

```
'FA-NAME'     = 'terminal'
'SEED-REGION' = 'terminal-building'
'DEFINITION'  = 'parking-lot
                && parking-apron && road'
```

The SEED-REGION attribute forces the interpretation system to create terminal functional-areas only if a terminal-building hypothesis exists that is consistent with one or more hypotheses of the types occurring in the DEFINITION attribute.

Model-Generation

The knowledge embedded in the model-generation phase has to do with using the context in which a particular region is found to determine which of several conflicting interpretations are correct. Commonly occurring conflicts can be enumerated, and more expensive knowledge-intensive operators can be applied to resolve these conflicts in the context of a particular scene model. The general syntax of a model-generation schema looks as follows:

```
'CONFLICT'   = '<hypothesis1> && <hypothesis2>'
'RESOLUTION' = '<keyword> [&& <keyword-data>]'
        <any number of resolutions>
```

For example, consider the following schema:

```
'CONFLICT'   = 'hangar-building && parking-lot'
'RESOLUTION' = 'function && stereo'
```

This schema will invoke a stereo operator to decide whether or not a region has height, so that the interpretation system can decide between the hangar-building or the parking-lot hypothesis.

<keywords>	<keyword-data>
default	none
function	name of function used to do resolution
conclusion	name of function used to combine results

If there is more than one resolution specified, then there must be a conclusion resolution specified. The conclusion will take the results of all of the resolution strategies and determine what the final result will be.

Appendix II

Some examples of the productions generated by RULEGEN are now given. Because the high-level rule descriptions were given along with the schemata in the previous appendix, here we will attempt to describe how the productions actually implement semantics of each rule.

Region-to-Fragment

Using the example schema for the region-to-fragment phase given in Appendix I, the system generated OPS5 productions defining the 'house' subclass. The first production finds an uninterpreted region in working-memory, and sets up a subtask which constrains OPS5 conflict-resolution to the productions in the given group only. These other productions apply the geometric constraints and leave the results of each test in a special LISP data-structure. Finally, domain-independent productions finalize this process by doing the final test evaluations, deciding whether or not an interpretation should be created, and removing the now obsolete subtask.

```
(p RTF::HS::initialize-HS-attributes
    (rtf-task ^region <name> ^data <token>)
    (region ^symbolic-name <name> ^house nil)
    (fragment ^symbolic-name <name>
        ^object-type compact
        ^hypothesis unknown)
  -->
    (make rtf-subtask ^ruleset HS::match-HS-attributes
        ^region <name> ^data <token> house)
)

(p RTF::HS::match-HS-area
    (rtf-subtask ^ruleset
        { <ruleset> = HS::match-HS-attributes }
        ^region <name> ^data {} <hyp>)
  { (rtf-rule-constants ^ruleset <ruleset>
        ^attribute area) <constants> }
    (region ^symbolic-name <name> ^area <value>)
  -->
    (bind <index> (litval constants))
    (call OPS::match-score <name> <hyp> <value>
        (substr <constants> <index> inf))
)

(p RTF::HS::match-HS-ellipse-length
    (rtf-subtask ^ruleset
        { <ruleset> = HS::match-HS-attributes }
        ^region <name> ^data {} <hyp>)
  { (rtf-rule-constants ^ruleset <ruleset>
        ^attribute ellipse-length) <constants> }

      (region ^symbolic-name <name>
          ^ellipse-length <value>)
    -->
      (bind <index> (litval constants))
      (call OPS::match-score <name> <hyp> <value>
          (substr <constants> <index> inf))
  )
```

```
(p RTF::HS::match-HS-ellipse-width
    (rtf-subtask ^ruleset
        { <ruleset> = HS::match-HS-attributes }
        ^region <name> ^data {} <hyp>)
  { (rtf-rule-constants ^ruleset <ruleset>
        ^attribute ellipse-width) <constants> }
    (region ^symbolic-name <name>
        ^ellipse-width <value>)
  -->
    (bind <index> (litval constants))
    (call OPS::match-score <name> <hyp> <value>
        (substr <constants> <index> inf))
)

(p RTF::HS::match-HS-ellipse-linearity
    (rtf-subtask ^ruleset
        { <ruleset> = HS::match-HS-attributes }
        ^region <name> ^data {} <hyp>)
  { (rtf-rule-constants ^ruleset <ruleset>
        ^attribute ellipse-linearity) <constants> }
    (region ^symbolic-name <name>
        ^ellipse-linearity <value>)
  -->
    (bind <index> (litval constants))
    (call OPS::match-score <name> <hyp> <value>
        (substr <constants> <index> inf))
```

Local-consistency

Another example schema from Appendix I, for the local-consistency phase of SPAM, produces a set of productions defining the spatial relationship constraining houses to be parallel to roads. The first two productions, `call` and `init`, establish a subtask which, again, constrains the conflict-resolution process to the current production group. The next two productions, `invalid-type` and `no-rule-constraints`, do error checking. The next production, `exit`, removes the current subtask so that the remaining local-consistency rules can fire. The next two productions, `choose-RD` and `stop-choosing`, implement a loop in OPS5, so that all the computations can be performed at one time. At this point, domain-independent control productions take over and coordinate the spawning of sub-processes to do the low-level spatial calculations.

When these processes have completed, the results are placed into working memory and control is allowed to pass back to this production group. Finally, the last two productions, `satisfied` and `unsatisfied`, will match this result data and create subtasks that will be used by domain-independent productions to update confidences appropriately.

```
(p LCC::houses-are-parallel-to-roads::*call*
    (consistency-task
        ^hypothesis house ^fragment <id>
        ^region <name> ^misc <con>)
  -->
    (make lcc-subtask
        ^rulename HS::houses-are-parallel-to-roads
        ^hypothesis house ^fragment <id>
        ^region <name> ^misc <con>)
```

```
(p LCC::houses-are-parallel-to-roads::*init*
  { (lcc-subtask
      ^rulename
    { <rulename> = HS::houses-are-parallel-to-roads }
      ^hypothesis house ^fragment <id> ^region <name>
      ^misc <con>) <subtask> }
   (lcc-rule-constants ^rulename <rulename>)
  -->
   (call OPS::dumpstate)
   (remove <subtask>)
   (make lcc-rule-set ^rulename <rulename>
       ^hypothesis house ^fragment <id>
       ^region <name> ^misc <con>)
   (make lcc-chain ^rulename <rulename>
       ^taskname start-choose-mode)
)

(p LCC::houses-are-parallel-to-roads::*invalid-type*
  { (lcc-subtask
      ^rulename
    { <rulename> = HS::houses-are-parallel-to-roads }
      ^hypothesis { <hyptype> <> house }) <subtask> }
  -->
   (remove <subtask>)
   (write (crlf) (tabto 9)
       <rulename> -- Invalid hypothesis
       <hyptype> for this ruleset.
       (crlf))
)

(p LCC::houses-are-parallel-to-roads::*no-rule-constant
  { (lcc-subtask
      ^rulename
    { <rulename> = HS::houses-are-parallel-to-roads }
      ^hypothesis house) <subtask> }
  - (lcc-rule-constants ^rulename <rulename>)
  -->
   (remove <subtask>)
   (write (crlf) (tabto 9)
       <rulename> -- No rule constants
       for this ruleset.
       (crlf))
)

(p LCC::houses-are-parallel-to-roads::*exit*
  { (lcc-rule-set
      ^rulename HS::houses-are-parallel-to-roads)
    <ruleset> }
  - (geometry)
  - (queue)
  -->
   (remove <ruleset>)
)

(p LCC::houses-are-parallel-to-roads::*choose-RD*
   (lcc-chain ^rulename
     { <rulename> = HS::houses-are-parallel-to-roads }
       ^taskname start-choose-mode)
   (lcc-rule-set ^rulename <rulename> ^region <name0>
       ^fragment <id0> ^misc <conf0>)
   (fragment ^lcc-participant yes ^hypothesis road
       ^symbolic-name { <name1> <> <name0> }
       ^fragment-token <id1> ^confidence <conf1>)
   (lcc-rule-constants ^rulename <rulename>
       ^constants <thresh0-1> {})
  -->
   (call OPS::queue-task orientation parallel <name0>
       <name1> <thresh0-1> <id0> <conf0> <id1> <conf1>)
)
```

```
(p LCC::houses-are-parallel-to-roads::*stop-choosing*
  { (lcc-chain ^rulename
     { <rulename> = HS::houses-are-parallel-to-roads }
       ^taskname start-choose-mode) <chain> }
    (lcc-rule-set ^rulename <rulename>)
  -->
    (remove <chain>)
)

(p LCC::houses-are-parallel-to-roads::*satisfied*
    (lcc-rule-set
       ^rulename
     { <rulename> = HS::houses-are-parallel-to-roads }
       ^fragment <id0>)
    (lcc-rule-constants ^rulename <rulename>
       ^constants <min> <max>)
  { (geometry ^type orientation ^subtype parallel
       ^frag1 <idT> ^con1 <cT> ^frag2 <id> ^con2 <c>
       ^values { <value> >= <min> <= <max> })
    <geometry> }
    (fragment ^fragment-token <id0>)
  -->
    (remove <geometry>)
    (bind <score>
       (OPS::geometric-score 0.0 <threshold> <value>))
    (bind <eltlen> 3)
    (make lcc-updates ^rulename <rulename>
         ^elt-len <eltlen> ^fragment <idT>
         ^data <id> <c> <score>)
    (make lcc-updates ^rulename <rulename>
         ^elt-len <eltlen> ^fragment <id>
         ^data <idT> <cT> <score>)
```

```
(p LCC::houses-are-parallel-to-roads::*unsatisfied*
   (lcc-rule-set
      ^rulename
   { <rulename> = HS::houses-are-parallel-to-roads })
   (lcc-rule-constants ^rulename <rulename>
      ^constants <threshold>)
{ (fragment ^fragment-token <idT> ^test-count <count>)
     <fragment> }
{ (geometry ^type orientation ^subtype parallel
      ^frag1 <idT>) <geometry> }
   -->
   (modify <fragment>
      ^test-count (compute <count> + 1))
   (remove <geometry>)
)
```

Appendix III

Figure 1 is an example of the output generated by SPATS for the region-to-fragment phase of SPAM. The explanations are generated as part of the final output for easy reference.

rom these statistics, one can see that the system was able to correctly interpret the linear and large-blob classes without any any misinterpretations at all. For the compact class, notice that the number of hangar building interpretations is identical to the number of compact

```
            Id: Moffett1

Column                          Explanation
------                          -----------
Class/Subclass: Class/Subclass to be analyzed
GndTth:         # of occurrences of the class/subclass in gnd truth table.
                Each class entry is the sum of its subclass entries.
WMEs:           # of region WMEs whose symbolic names match the subclass's
                gnd truth IDs and that have subclass or class interps.
                NOTE: The difference between the class entry and the sum
                of its subclass entries is the # of region WMEs with only
                class interps.
CorrWMEs:       # of the aforementioned WMEs which contained the
                correct subclass or class interpretation.
IncorrWMEs:     # of the aforementioned WMEs which did not contain
                the correct subclass or class interpretation.
                NOTE: The sum of the CorrWMEs and IncorrWMEs entries for
                each class or subclass should add up to its WMEs entry.
CorrBF:         The branching factor for the correct interpretations
                of the subclass or class. The BF shows how many
                interpretations, there were, on the average, for each correct
                interpretation. If a class/subclass has a CorrBF of 0, then
                it had no correct interpretations.
IncorrBF:       The branching factor for the incorrect interpretations
                of the subclass or class.

Class/subclass          GndTth  WMEs  CorrWMEs  IncorrWMEs  CorrBF  IncorrBF
   linear                 40     40      40          0       6.65     0.00
       runway              2      2       2          0       3.00     0.00
       taxiway            35     36      36          0       6.89     0.00
       road                2      2       2          0       6.00     0.00
   compact                 9      9       5          4       7.00     6.00
       hangar-building     9      9       5          4       7.00     6.00
       terminal-building   0      0       0          0       0.00     0.00
   small-blob              6      6       4          2       8.50     4.00
       parking-apron       3      3       1          2       7.00     4.00
       parking-lot         3      3       3          0       9.00     0.00
   large-blob             12     12      12          0       5.92     0.00
       grassy-area        11     11      11          0       6.09     0.00
       tarmac              1      1       1          0       4.00     0.00
Final Stats:              67     67      61          6       6.66     5.33
```

Figure 1: Example SPATS output for region-to-fragment phase.

interpretations. This shows us that the geometric constraints for the subclass hangar-building are not discriminatory enough. This shows up in the correct branching factor as well.

An example of the output for the functional-area phase is given in figure 2. This summarizes all the correct and incorrect hypotheses participating in the created functional-areas. We can use this information to determine the status of the high-level groupings generated by SPAM. If it is recognized that many incorrect interpretations are being used to support correct interpretations (or visa-versa) when creating a functional-area, then the local-consistency knowledge is at fault, as it is not properly characterizing the spatial layout of the scene.

```
    Id: Moffett1

Functional Area Type: All functional areas
Functional Area ID: All functional areas
Total # of functional-areas: 83
Total # of fragments (from FA, consistent-fragments,
                            and inconsistent-fragments lists): 60
Total # of the above fragments found in ground truth file: 60
Fragments composing FA(s):
        # of correct fragment hypotheses:     60
        # of incorrect fragment hypotheses:   17
Consistent-Fragments:
        # of correct fragment hypotheses:      1
        # of incorrect fragment hypotheses:    4
Inconsistent-Fragments:
        # of correct fragment hypotheses:      1
        # of incorrect fragment hypotheses:   15
Correct fragment hypotheses table:
                               Ground Truth Types
                   RW     TW     RD     HG     TB     PA     PL     GA     TM
        RW          2      0      0      0      0      0      0      0      0
        TW          0     38      0      0      0      0      0      0      0
        RD          0      0     23      0      0      0      0      0      0
Frag.   HG          0      0      0      3      0      0      0      0      0
Types   TB          0      0      0      0      0      0      0      0      0
        PA          0      0      0      0      0      6      0      0      0
        PL          0      0      0      0      0      0     13      0      0
        GA          0      0      0      0      0      0      0     27      0
        TM          0      0      0      0      0      0      0      0      0

Incorrect fragment hypotheses table:
                               Ground Truth Types
                   RW     TW     RD     HG     TB     PA     PL     GA     TM
        RW          0      4      1      0      0      0      2      0      1
        TW          0      0      0      0      0      0      1      0      0
        RD          0     86      0      5      0      0      5      0      2
Frag.   HG          0     19      0      0      0      0      1      0      0
Types   TB          0     73      5     12      0      0      1      0      6
        PA          0     13      0     11      0      0      3     38      9
        PL          0     86      0     39      0      9      0     63      2
        GA          0     14      0     11      0      3      6      0      1
        TM          0      0      0      4      0      1      1      6      0
```

Figure 2: Example SPATS output for functional-area phase.

USING GENERIC GEOMETRIC MODELS FOR INTELLIGENT SHAPE EXTRACTION

Pascal Fua and Andrew J. Hanson *

SRI International (Artificial Intelligence Center)
333 Ravenswood Avenue, Menlo Park, California 94025

Abstract

Object delineation based only on low-level segmentation or edge-finding algorithms is difficult because typical edge maps have either too few object edges or too many irrelevant edges, while object-containing regions are generally oversegmented or undersegmented. We correct these shortcomings by using model-based geometric constraints to produce delineations belonging to generic shape classes. Our work thus supplies an essential link between low-level and high-level image-understanding techniques. We show representative results of applying our models for buildings, roads, and trees to aerial images.

1 INTRODUCTION

Our goal is to delineate probable instances of generic object types in images of intermediate resolution. Such images have resolution adequate for humans to perceive shapes clearly, but not so fine that small details and textures would dominate the description given by a human observer. In Figure 1, we present a typical aerial image of this class that contains a combination of suburban features, along with a corresponding edge image [Canny, 1986] and a segmentation [Laws, 1984].

A standard low-level approach to the task of extracting objects such as buildings from Figure 1a would attempt to match region boundaries or edge groups with the edges of a building template. However, when we examine the data, we see that neither regions nor edges correspond reliably to building objects. The segmentation boundaries tend either to break a building roof into pieces or to merge extraneous areas with those identifiable as roofs. The Canny edges, on the other hand, do not include several critical edges in the center building or the road, even though these are extracted as region boundaries by the segmentation.

Clearly, no single parameter setting for conventional segmentation or edge-finding techniques can be expected to handle all target objects in one image, much less in multiple images. Additional information must therefore be provided in order to generate object delineations that are sufficiently reliable to be useful for applications such as context-specific labeling systems [see, e.g., Brooks, 1981; McKeown et al., 1985].

The key elements of our approach to solving this problem are the following:

This research was supported in part by the Defense Advanced Research Projects Agency under Contract No. MDA903-86-C-0084.

- **Define Generic Shape Models.** We avoid the drawbacks of rigid template models and produce delineations that are not necessarily tied to any specific labeling scheme by defining shape models for generic classes of objects. When we supplement low-level data with the predictive power of such models, we are able to recover information that is more likely to be semantically meaningful.
- **Integrate Edge-Based and Area-Based Geometric Constraints.** Both the edges and the areas of a feature contain geometric information relevant to the task of identifying it as an instance of a generic model. We use edges to generate overall geometry and to provide estimated area outlines. Areas that are associated with edges are tested for compatibility with the object model and with one another; we use the RANSAC random sample consensus technique [Fischler and Bolles, 1981] to compute optimal model fits that systematically discount gross anomalies.
- **Predict and Verify of Model Components.** Missing components of models are predicted and checked using an adaptive search procedure; our implementation uses a gradient ascent method [Leclerc and Fua, 1987] to search for predicted edges. Thus, for example, we can reconstruct and locate building boundaries and road edges that might be unrecoverable using conventional methods; an edge-detector parameter setting weak enough to find such missing edges at the beginning would yield an edge-map dominated by irrelevant noise.

2 GENERIC MODELING

People can accurately classify instances of various object categories even though a particular instance may have a unique shape that they have never seen before. Generic shape classes provide a good approach to automating this human ability. Generic models that we have found useful for analysis of *real images* possess the following characteristics:

- **Strong edge geometry.** The elementary edge or line data extractable from an image must be related in some direct and computable way to the object. In particular, the model must suggest explicit rules for dealing with anomalies and predicting likely locations of missing geometrical components. Typical models include edge geometry characterized by long, straight, line edge segments, by edges or lines with uniform local curvature, and by edges with a good statistical signatures characterizing their jaggedness. In addition, there must be mechanisms for the production of coherent area-enclosing structures. Thus, for example, parallel edges, corners, equidistant curved lines, and edges outlining a compact shape are reasonable geometric substructures that can be used to delineate areas that are portions of the larger structure.

- **Strong area signature.** Areas contained within a substructure of a generic object should be characterizable by a computable signature. If anomalies are expected, they should be clearly distinguishable using the area signature and should ideally include no more than a small fraction of the area. (Examples of such areas are parking lots with cars or roofs with chimneys.) Typical area signatures would be the presence of uniform or uniformly changing intensity values or textures. Anomalies in such a background are easily located and discounted using a RANSAC procedure to fit planes to the intensity values within the delineated area.

The models that we have implemented – buildings, roads, and trees – contain the following universal components: (1) Edge definition, (2) Composite structure definition, (3) Linking geometry specification for composite structures, (4) Area signature specification, and (5) a Geometric completion model. The components of each of these models are are summarized in Table I. The most general model-parsing procedure that we have needed to interpret each of these models in an image includes the following elements:

1. **Build the edges according to the edge definition.**
2. **Construct composite structures from the edges.**
3. **Construct test areas using the enclosing geometry of the composite structures, and group structures with consistent linking geometry and area signature.**
4. **Predict and search for missing elements of the model geometry.**
5. **Fill in remaining boundary gaps to make a complete delineation.**
6. **Compare the resulting delineation to the characteristics of the original model.**

The overall approach can clearly be extended to any other object for which appropriate characteristics can be formulated, e.g, cylindrical oil tanks, drainage patterns, and buildings with perspective distortion.

In the following subsections, we outline the features of our models for buildings, roads, and trees, and illustrate how these models fit into the general framework. Where space allows, we mention some details of the individual requirements of the model parsing framework outlined above.

2.1 Buildings – Rectilinear Networks

Our most extensive work so far has been devoted to the task of delineating rectilinear, presumably cultural, structures [Fua and Hanson, 1985, 1986].

We characterize buildings and related cultural structures (e.g., parking lots, patios, gardens, and courtyards) as rectilinear networks of adjacent or joinable area-enclosing straight-edge groups enclosing areas with planar intensity.

The basic parsing procedure for generic rectilinear structures follows the pattern given above. Since region boundaries of a histogram-based segmentation [Laws, 1984; Ohlander et al. 1978] tend to correspond to high image gradients, the straight edges are extracted as sequences of pixels with consistent gradient directions. While single segmentations often have inadequate characteristics, segmentations with increasingly permissive parameters produce regions that are first undersegmented, then well segmented and finally oversegmented as shown in Figure 2. The multiple data sources lead to a network of geometrically consistent straight edges, shown in Figure 2f, which are used to drive the geometric processes.

In practice, region boundaries may be off by a few pixels from the actual edge location; we optimize their locations using the gradient-ascent procedure. In each of the segmentation regions, edges that are parallel or perpendicular are singled out for special consideration. These associated edges, together with the region they come from, define areas in the image. Areas are tested for consistency with a RANSAC planar fit in intensity space, and edges that generate qualifying areas are retained for further parsing.

We note that the same edge can belong to several structures. If the structures are compatible with respect to the structure linking specification and enclosed area characteristics, new geometric relationships between edges, such as collinearity, are instantiated. The result is that edges are grouped into networks defined by graphs of the geometric relations among them.

Rectilinear geometric relationships are used to predict how the edges should be linked and where missing edges might be. The predictions are fed to the adaptive straight-edge finder [Leclerc and Fua, 1987] or to the F^* edge finder [Fischler et al., 1981] if a straight edge link is not found.

The networks of compatible composite structures are then connected to form closed contours and define new semantically motivated regions that are the final output of the current system. The candidate features can be scored using a measure of the closeness of the delineation characteristics to those expected in an ideal model instance.

In practice, the information required to assign meaningful labels to candidate cultural structures can be very primitive; we will give some examples below in which even such simple techniques as clustering based on region-similarity measures are quite effective.

2.2 Roads – Curvilinear Parallel Networks

It is straightforward to modify the rectilinear cultural feature model to include smoothly curving road segments. The edges of such roads are almost straight in most places and can be detected locally using the techniques given above. The edges are then grouped into parallel structures and linked into elongated networks that may have large-scale curvature. To deal with winding roads, the straight edges can be replaced by smoothly curved edges while retaining the rest of the approach. (See Table I for a summary.)

The only major change in the road model is the rule used to predict missing components of the geometric structure. First, the initial network of parallel edges is used to estimate the location of the center of the road and its width. Next, we fit a spline to the estimated center of the road and use it to define two parallel splines that correspond to its edges. Using the gradient ascent method, we optimize the location of the two splines under the constraint that they must remain parallel. This is a powerful technique because wherever one side of the road is lost due to poor photometry or occlusions, the edge information present on the other side can still be used to guide the optimization procedure.

2.3 Trees – Irregular Clumps

Vegetation clumps, typically small groups of trees, are characterizable as being complementary to the regular cultural-object models we have described so far. Their edges are typically jagged and irregular, so any compact object that has no components that are road-like or building-like could be a candidate for vegetation. Other irregular objects such as rock outcroppings, bodies of water, and drainage patterns would have similar signatures. The tree model is summarized in Table I.

The parsing procedure for vegetation clumps first identifies the jagged edges bordering an area with consistent signature, and then uses F^* to connect the edges along the path with strongest image gradient.

3 TYPICAL RESULTS

In this section we present some representative results of applying our approach to aerial imagery.

To illustrate the behavior of the system on buildings, we have chosen two images that are especially challenging in terms of shape complexity and faint edge photometry. In Figure 3, we show the results of analyzing the image shown in Figure 1a. Figure 3a shows the initial set of networks, selected in this case on the basis of a size filter; if we add a selection criterion based upon clustering areas with similar intensity characteristics, one of the clusters is the set of house candidates in Figure 3b.

Figure 4a shows another example of an image containing difficult-to-parse cultural structures; in particular, note the extreme weakness of many relevant roof edges. Figure 4b shows a cluster of bright enclosures that can be identified as sunlit roofs, Figure 4c shows a corresponding cluster of shaded roof sections, and Figure 4d gives the complete composite roof structures.

Turning our attention now to linear features, we take the same image shown in Figure 1a and apply the model for generic road segments. The system finds the initial set of straight edges shown in Figure 5a, groups them into equidistant parallels, connects those that seem to be collinear or smoothly curving, and uses them to predict the approximate delineation of the road as shown in Figure 5b. Finally, the predicted shape is optimized with respect to variations in the global width and local curve skeleton, thus yielding Figure 5c.

Finally, we apply the parsing procedure to vegetation clumps. In Figure 6a, we show an image containing typical vegetation clumps, along with one of a set of segmentations in Figure 6b. The initial candidates for vegetation clumps are shown in Figure 6c, with a final selection filtered on image intensity in Figure 6d.

4 CONCLUSIONS

In this work, we have proposed an approach based on generic models and a combination of edge-driven and photometry-based geometric reasoning to delineate several classes of objects in aerial images. Such delineations may be utilized in a variety of ways, but are especially appropriate as input to high-level knowledge-based systems. Since the discovered shapes are generic, there is no *a priori* commitment to a particular labeling or modeling system.

We have devised methods for

- **Integration of Multiple Geometric Data Sources.**

  Data-driven edge-extraction and image-segmentation processes do not perform well on multiple target objects. We combine multiple information sources and use both edge geometry and enclosed area characteristics to generate and verify shape hypotheses; we thus make efficient use of the available geometric information in the image.

- **Generic Shape Extraction.**

  For many important tasks, the exact shapes of objects of interest are not known. We define and use generic models to deal with whole classes of objects. Within the context of such models, we recover expected but missing model components using adaptive search techniques, and compensate for photometric anomalies. In particular, we have proposed models for cultural structures, roads, and vegetation clumps, all of which fit into a universal format for model definition and parsing.

The system's effectiveness derives from the definition and use of generic shape models to refine and interpret low-level image information. The clear delineations that we can produce are essential for application-oriented parsing schemes, and provide an adequate basis for rule-based labeling systems that could not function with traditional low-level data alone.

REFERENCES

R.A. Brooks, "Symbolic Reasoning Among 3-D Models and 2-D Images," Artificial Intelligence Journal **16** (1981).

J. Canny, "A Computational Approach to Edge Detection," IEEE Trans. PAMI **8**, pp. 679–698 (1986).

M.A. Fischler and R.C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," CACM, **24**, pp. 381–395 (June 1981).

M.A. Fischler, J.M. Tenenbaum, and H.C. Wolf, "Detection of Roads and Linear Structures in Low-Resolution Aerial Imagery Using a Multisource Knowledge Integration Technique," Computer Graphics and Image Processing **15**, pp. 201–223 (1981).

P. Fua and A.J. Hanson, "Locating Cultural Regions in Aerial Imagery Using Geometric Cues," Proceedings of the Image Understanding Workshop, pp. 271–278 (December 1985).

P. Fua and A.J. Hanson, "Resegmentation Using Generic Shape: Locating General Cultural Objects," Pattern Recognition Letters (1986) *in press*.

K.I. Laws, "Goal-Directed Texture Segmentation," Technical Note 334, Artificial Intelligence Center, SRI International, Menlo Park, California (September 1984).

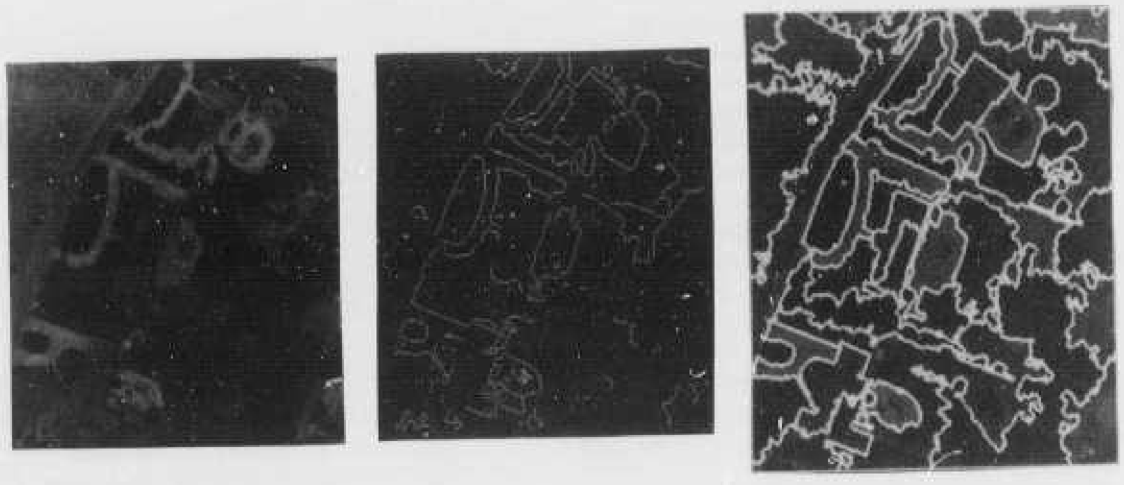
Y. Leclerc and P. Fua, "Finding Object Boundaries Using Guided Gradient Ascent," in this Proceedings.

D. McKeown, W.A. Harvey, and J. McDermott, "Rule-Based Interpretation of Aerial Imagery," IEEE Trans. PAMI **7**, pp. 570–585 (1985).

R. Ohlander, K. Price, and D.R. Reddy, "Picture Segmentation Using a Recursive Region Splitting Method," Computer Graphics and Image Processing **8**, pp. 313–333 (1978).

Model Component	**Buildings**	**Roads**	**Trees**
Edge definition	Straight	Curved	Jagged
Composite structure definition	Parallel and perpendicular	Parallel	Cluster
Linking geometry specification for composite structures	Rectilinear	Curvilinear	Free form
Area signature specification	Planar intensity	Planar intensity	Planar intensity
Geometric completion model	Straight edge search	Curved edge search	Connecting path search

Table I. Summary of the characteristics of each of three models described in the text.



(a) (b) (c)

Figure 1: (a) A typical aerial image with suburban features. (b) A Canny edge map. (c) A Laws histogram-based segmentation.

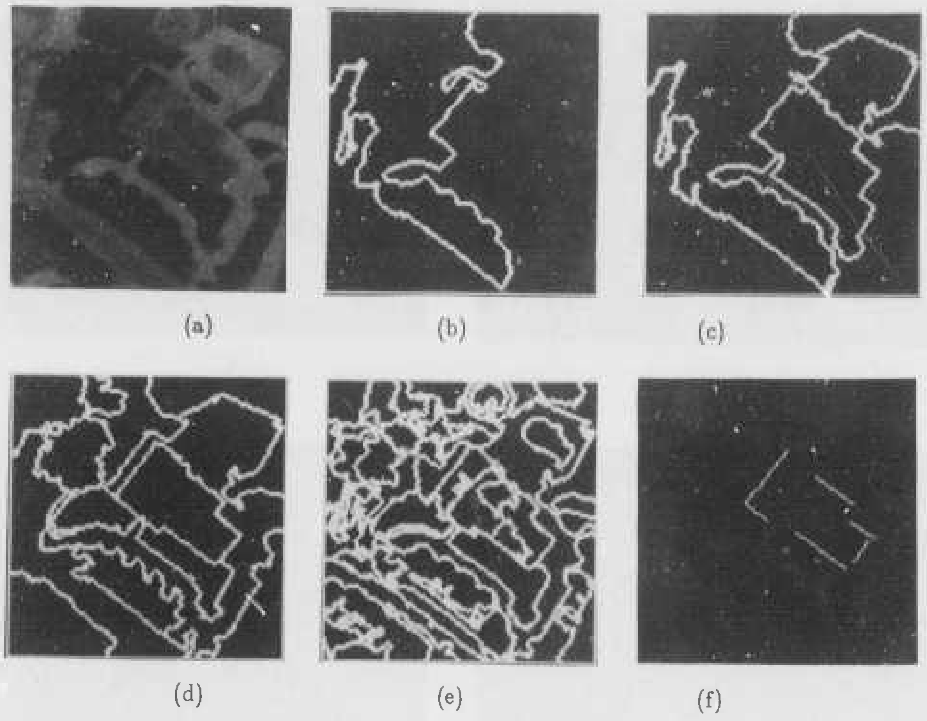


Figure 2: (a) A small image portion containing a cultural structure. (b) An extreme undersegmented partition. (c) An undersegmented partition. (d) An optimal partition for detecting the structure. (e) A highly oversegmented partition. (f) The set of long, straight edges extracted from the partition boundaries using the criterion that the edges enclose as large a uniform rectilinear area as possible. These edges form a network.

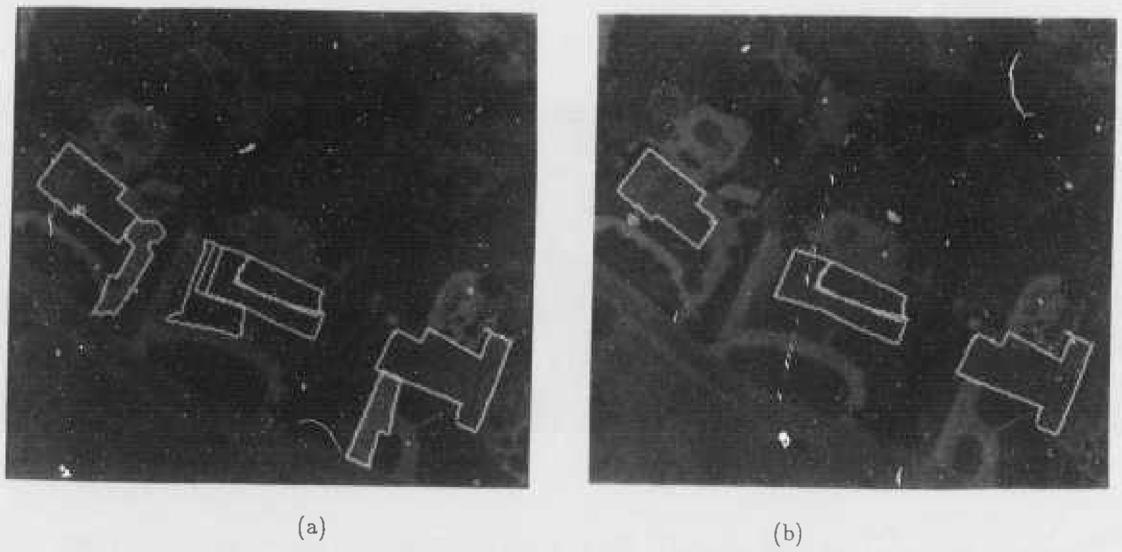


Figure 3: (a) Rectilinear networks meeting a size criterion. (b) House-like networks found by imposing in an additional region-uniformity filter.

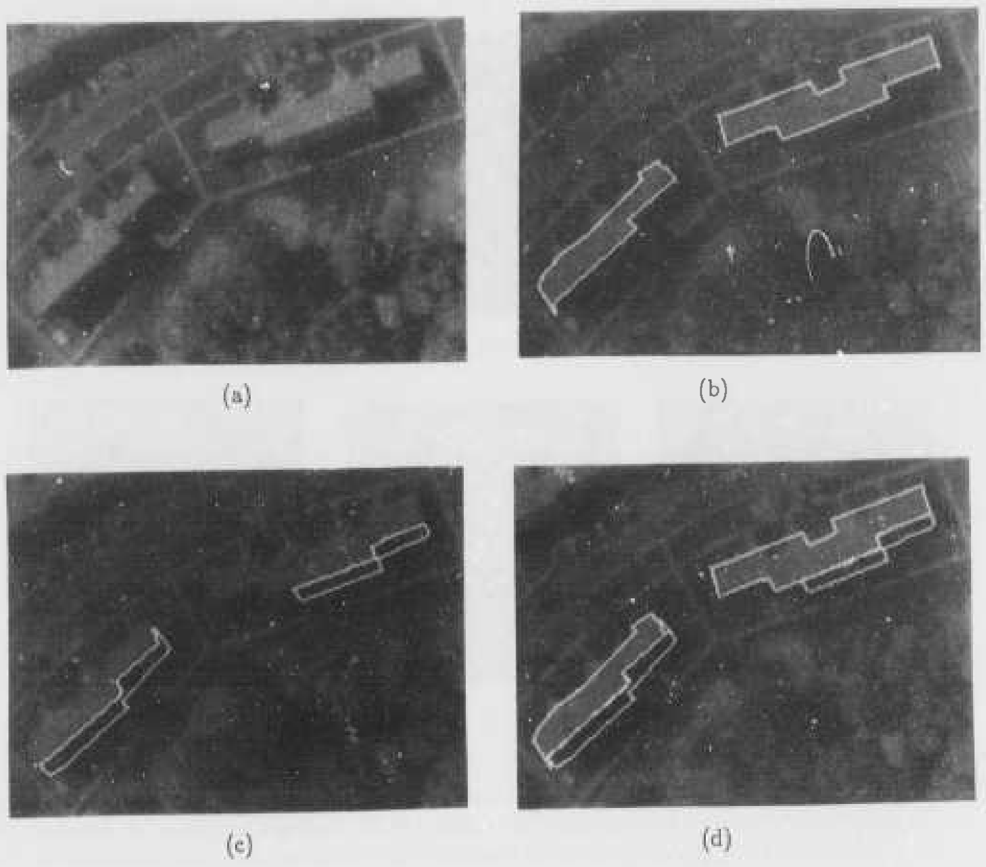


Figure 4: (a) An image containing complex buildings with some faint edges. (b) A sunlit roof cluster. (c) A shaded roof cluster. (d) House candidates constructed by merging the sunlit and shaded roof candidates.

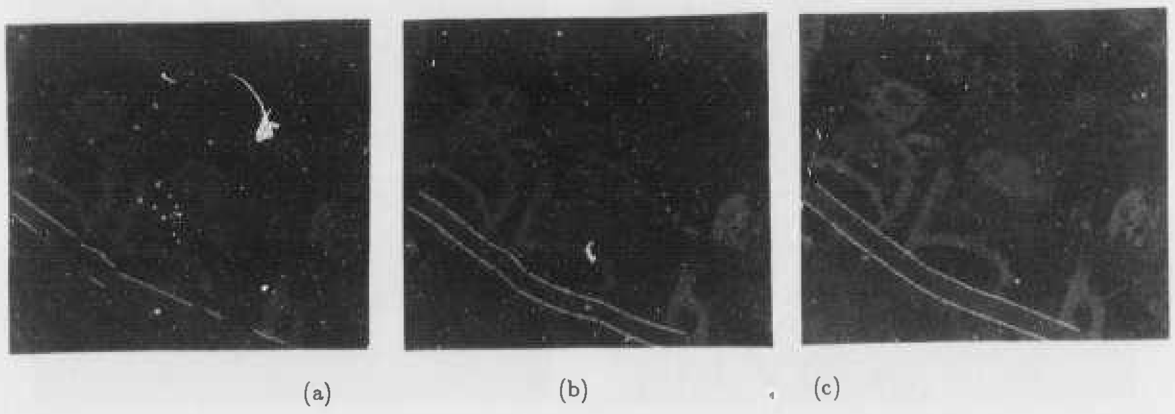


Figure 5: An example of a road segment. (a) The edges that are originally grouped together as a possible road structure. (b) Intermediate prediction of the road path given only the initial edges. (c) Final road position optimized to choose best path with the same (variable) width for the entire length.

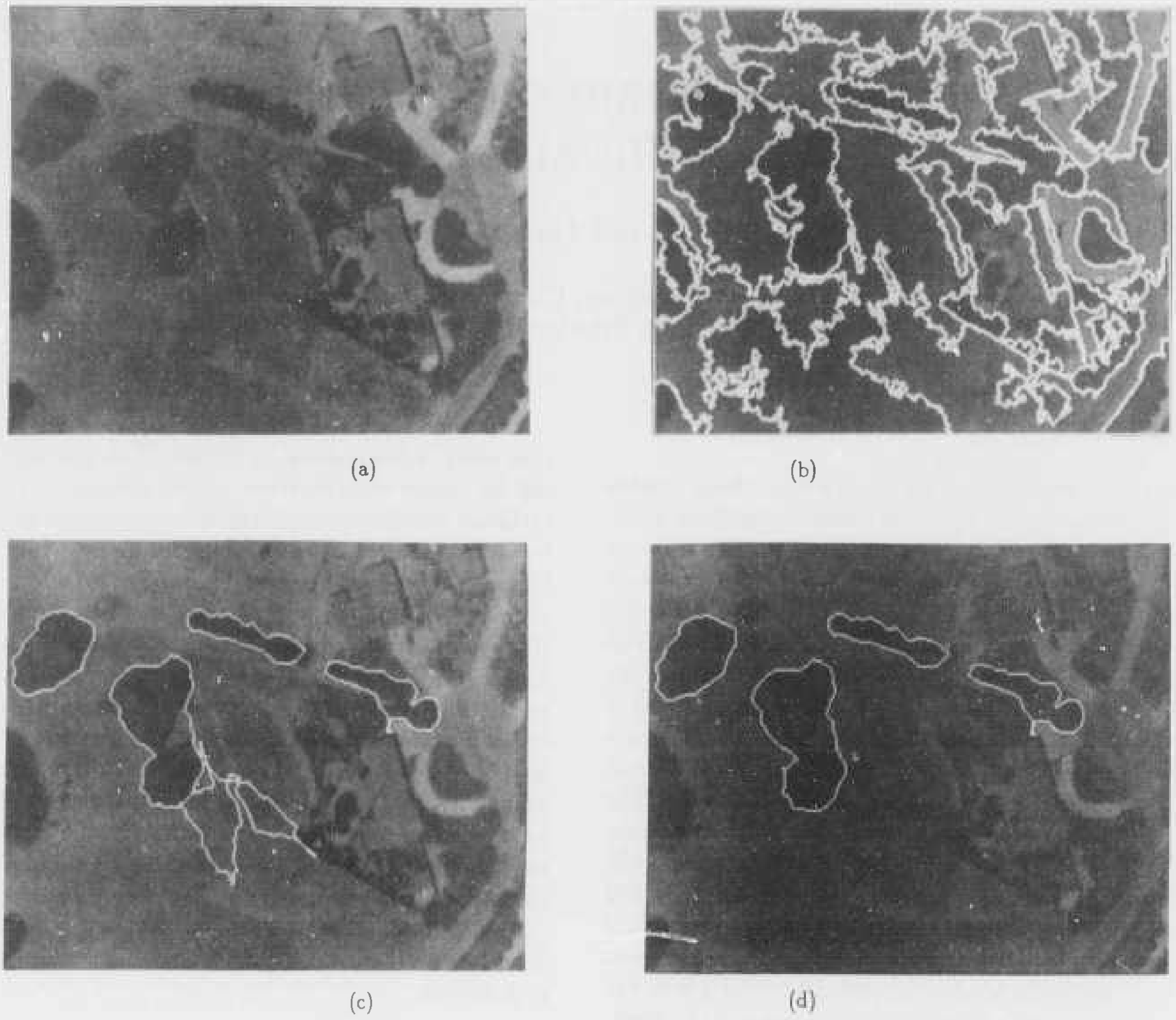


Figure 6: (a) An image containing vegetation clumps. (b) One of a family of segmentations used to derive edge candidates. (c) The initial set of vegetation clump candidates. (d) Vegetation candidates selected on the basis of the intensity of the enclosed area.

STEREO CORRESPONDENCE: A HIERARCHICAL APPROACH*

Hong Seh Lim and Thomas O. Binford

Artificial Intelligence Laboratory, Computer Science Department,
Stanford University, Stanford, CA 94305, U.S.A.

Abstract

This paper describes a high-level stereo vision system for recovering depth data of a three-dimensional scene from a stereo pair of gray-scale images. The features chosen for stereo matching are edgels (i.e., short, linear edge-elements, each characterized by a direction and a position), junctions, curves, surfaces, and bodies. The system first performs a monocular interpretation on the scene in each image. It builds a hierarchical structure of the scene from the chosen features. Low-level features constitute the lower layers of this hierarchical structure, while high-level features form the higher layers. The system starts matching features between the hierarchical structures at the highest level. Results from the matching of high-level features are used to guide and constrain the matching of lower-level features. These guidance and constraints are propagated to the lowest level of the hierarchical structure. The results from the matching provide a segmented depth map of the scene.

This hierarchical approach enables a globally consistent matching result and avoids local mismatches. It reduces searching time for locating correspondence features during the matching processes. The resulting depth map is segmented and readied for surface interpolation. We also discuss how to utilize the constraint from limbs for surface reconstruction.

1 Introduction

Extraction of depth information is one of the most important processes in the visual understanding of images. Among various approaches for extracting depth information, stereo matching has wide applications. It has the advantage of being a passive technique, requiring no active sensor. It has been applied to cartography [22], surveillance [10], passive navigation [7,17], industrial automation [20], and modeling [24].

*This work was supported by the Defense Advanced Research Projects Agency under contract number N00039-84-C-0211

This paper describes a high-level stereo vision system using a hierarchical approach. The features chosen for stereo matching are edgels, junctions, curves, surfaces, and bodies. A hierarchical structure of these features is built from each image before matching (see figure 1). Low-level features, i.e., edgels, junctions, and curves constitute the lower two layers of the structure. Higher-level features, i.e., surfaces and bodies, form the higher two layers. The system starts matching features at the highest level of the structure where the number of possible matches are small and mismatches are few. The results from these matches at this level are propagated to the next lower level, and they are used as guidance and constraints for matching at the lower levels. The propagation of constraints and guidance ends at the lowest level where edgels are being matched.

The following section briefly reviews past work. Then we present our approach and show examples of real images.

2 Previous Work

Two principal techniques have been used in stereo matching: area-based matching [9,22,17] and feature-based matching [3,21,16].

2.1 Area-based Matching

Area-based matching attempts to match small windows in each image by correlating their intensities. Ideally one would like to achieve pixel correspondence for each pixel in the window. But the information in one pixel is not enough to resolve the ambiguity in matching.

Area-based matching has been applied with some success in the analysis of aerial images, where the terrain varies smoothly and continuously. The presence of detectable texture is required for area-based matching. Area-based matching tends to break down when the image of the scene has textureless areas, repetitive

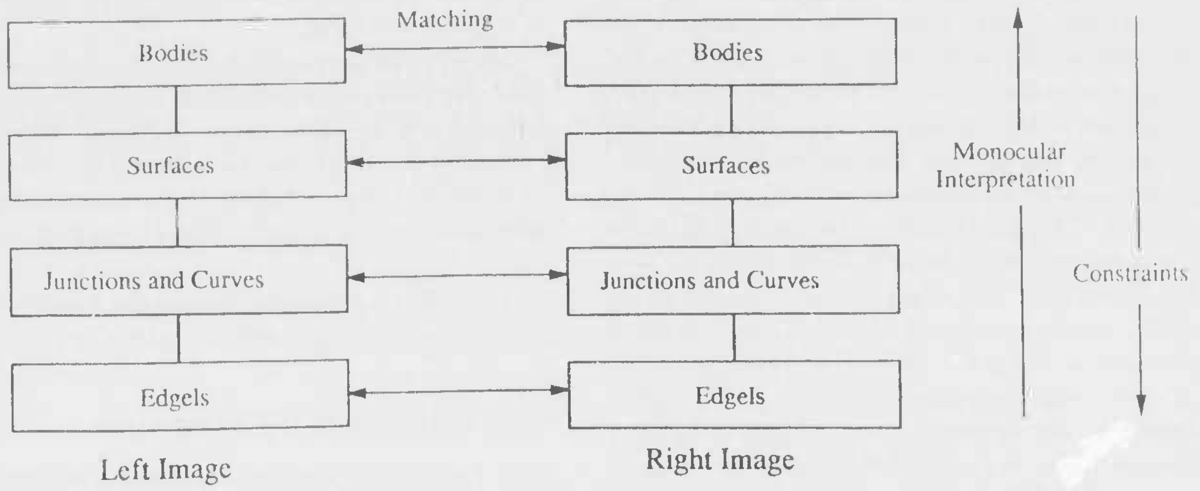


Figure 1: Hierarchical Structure

patterns, or surface discontinuities. For textureless areas, there are infinite matches. There can be multiple matches under repetitive patterns. At surface discontinuities, no correspondence between areas crossing the discontinuities is possible.

The accuracy for correspondence in area-based matching depends on the window size and is generally an order of magnitude less than that of feature-based matching. Computation time can be reduced by matching only areas that are of particular interest, e.g., with large variance.

2.2 Feature-based Matching

Feature-based matching uses features such as junctions, edgels or curves for matching. These features are extracted from the gray-scale image. They relate to intensity changes rather than raw intensities in the image. They provide a better characteristic of physical changes in the scene. The underlying principle is that discontinuities in the intensity represent discontinuities on the physical surfaces in the scene, except limb and cusp. The discontinuities can be caused by surface depth, orientation, reflectance, or illumination. All these discontinuities occur at points on the physical surfaces. By matching features derived from these discontinuities, we match physical points on the object surfaces. An exception arises at limbs of an object from different viewpoints since the limbs correspond to different points on the surface.

The location of the features in an image can be estimated to sub-pixel accuracy [11]. Consequently, the accuracy of the recovered three-dimensional depth is higher than that obtained by area-based matching. The number of features in an image is in general less than the number of pixels. As a result, the computation time for feature-based matching is less than that for area-based matching. Since not every point in an image corresponds to a feature, feature-based matching leads only to a sparse depth map. To produce a dense depth map, feature-based matching must be accompanied by model-based interpretation, surface interpolation, or by area matching.

The search space for matching features can be greatly reduced if one knows the geometric relationships between the cameras used in taking the pair of images. The family of planes passing through the two camera foci are called epipolar planes. Epipolar lines are the intersection of the epipolar planes with the image planes. Any image point lying on a particular epipolar line will find its corresponding points on the corresponding epipolar line in the other image.

In particular, if we restrict the camera geometry such that the principal horizontal lines of both images are collinear and the principal vertical lines of both images are parallel, (that is, the two cameras are related by a horizontal displacement), then epipolar lines are just horizontal lines in the image. The search for correspondence points will be limited to the same horizontal line. For images that are not registered in the epipolar geometry, locations of features can be transformed into the canonical stereo system [6].

Perkins [23] pointed out the difficulties involved in trying to resolve the matching problem without resort to higher-level information. There is little to characterize an edgel besides its location and orientation. A hierarchical scheme for matching stereo images of polyhedra was implemented by Ganapathy [5]. He also studied various rules for stereo matching. A cooperative computation algorithm was proposed by Marr, Poggio and Grimson [13,14,8] which matches random dot stereograms. It uses a uniqueness constraint to assign at most one disparity value to each point in the image and continuity constraint to require the disparity to vary smoothly, except at depth discontinuities. Arnold and Binford [1] used edge orientation, intensity and edge continuity to determine a set of globally optimal matches. They [2] also introduced quasi-invariants for correspondence of edges and surface normals. Mayhew and Frisby note that edges that lie on a continuous contour in one image should also have the continuity characteristic in the other image [15]. Baker and Binford [3] match edges on epipolar lines using those quasi-invariants and use dynamic programming to preserve the order of edges. A connectivity constraint was used to removed globally inconsistent edge correspondences. Ohta and Kanade [21] extended Baker's method from intra-scanline search to inter-scanline search. They took into account mutual dependency between epipolar lines in an image. Another system which uses segments, groups of collinear connected edge points, as matching primitive, was implemented by Medioni and Nevatia [16]. Its correspondence is based on a minimum differential disparity criterion.

3 Hierarchical Approach

Many stereo systems use low-level, local features, i.e., edgels and interest points, for matching. Because of the simplicity of these features, a local edgel or interest point in one image may match equally well with a number of edgels or interest points along epipolar in the other image. The problem is aggravated by the changes in the images of the corresponding features resulting from different viewpoints. These ambiguities in local matches can only be resolved by imposing global constraints.

Baker [3] incorporated intra-scanline constraints in the searching process and then checked continuity across scan-lines. Ohta [21] extended the method and included inter-scanline search in the process. Medioni [16] chose line segments as the matching features. All these techniques tried to incorporate inter-scanline connectivity constraints to ensure globally consistent matches.

Our system uses surfaces and bodies as matching features in addition to junctions, edgels, and curves. Surfaces and bodies are higher-level structures which give a less ambiguous representation of the underlying information in an image. A surface is bounded by a number of connected curves and junctions arranged in order. This ordering is a strong constraint for matching. Surfaces may be open or closed. A body is a collection of connected surfaces. Connectedness is another constraint for matching.

Ambiguities in matching of edgels can be resolved by the constraint of connected edgels across epipolar lines belonging to the same curve. Similarly, ambiguities in matching of curves can be resolved by the constraint of connected curves belong to the same surface. This also applies to connected surfaces belonging to the same body.

The following sections discuss the four levels of this hierarchical structure and the matching strategies.

3.1 Hierarchical Structure

The hierarchical structure is divided into four levels: bodies, surfaces, curves and junctions, and edgels. The lowest level of this structure consists of edgels. Connected edgels form curves at the next higher level. At this level junctions are found and classified appropriately. The next higher level consists of surfaces, which are formed by linking connected curves and junctions. Connected surfaces are in turn collected together to form bodies in the highest level of the hierarchical structure.

3.1.1 Edge Detection and Curve Fitting

A stereo pair of gray-scale images is obtained from a CCD camera. Edgels are detected by applying the Nalwa operator [18] to the pair of images. The operator fits a one-dimensional tanh-surface to each window in the image. An edgel is detected if the surface description is adequate in the least squares sense. Each edgel is characterized by direction and position. These edgels are first aggregated into ordered sets corresponding to individual extended edges. Curves are then fitted to the edgel-members of these edges in a best-fit sense [19]. The fitted curves are straight lines or conic sections.

3.1.2 Curve Extension and Junction Formation

The curves obtained by the curve fitting process are usually broken near junctions where three or more curves meet. The edge operator is designed to detect only one edge profile within each window. When more than one edge appear within the same window, the edge operator misses edges and estimates edge parameters inaccurately. Within a small disk around junctions, edge information is incomplete. However, the information of junctions is critical to the segmentation of surfaces and bodies.

The tangent of the fitted curve is a good indication of the orientation of the missing edges. Therefore a directional difference operator is chosen. This operator is applied to the region near the end points of each curve not connected to any junction. Contrast along the fitted curve guides the detection of step edges near junction. Thus the smaller operator operates directly on the underlying gray-scale image with guidance from the previously detected edgels and fitted curves.

The new junctions thus formed are classified by the type and order. The order indicates the number of curves belonging to the junctions. The end point of a curve which does not connect to any other curve will have an order of one. Junctions formed by two curves are invariably classified as *L* unless their tangent directions are opposite at the junction; then they are merged into one curve. Junctions with three curves are classified as either *A*, *T*, or *Y*. An *A* junction has one of its angles greater than 180 degrees. A *T* junction has any two of the three curves with opposite tangent directions. If all the angles between the curves are less than 180 degrees, then they form a *Y* junction. The *T* junction is evidence for an occlusion. This is a strong clue for segmentation [4,12]. We infer that the surface, formed by the top curve occludes the other surface. Junctions with higher order are not common, and they are classified as *X*.

3.1.3 Surface Segmentation

Surfaces are defined by boundaries of connected curves. Two kinds of surfaces are possible: open surfaces and closed surfaces. Open surfaces may result from an occluding surface or a surface with missing edges that are not detected because of noisy data.

Tracing surface boundaries is carried out by left-wall and right-wall followings. For left-wall following, the tracing starts at a junction, follows a curve, takes the left curve whenever it arrives at another junction. The process ends when it comes back to the starting junction or at a *T* junction or a junction of order one, depending on the initial conditions. For right-wall following, all the turns are to the right. When the tracing stops at a *T* junction or a junction of order one, the surface is an occluded or open surface.

3.1.4 Body Segmentation

Surfaces which share edges are grouped as bodies. Bodies form the highest level of the hierarchical structure. Each body thus obtained may consist of more than one object in the scene. The order of surfaces from left to right and top to bottom is noted. They are constraints for matching.

3.2 Matching Process

Two hierarchical structures of features are thus constructed from a pair of stereo images. The matching process starts matching at the highest level of these hierarchical structures, the body levels. Results from matching at this level are used to constrain and guide the matching at a lower level. We plan to match features in parallel at each level and utilize the results from a higher level to constrain and resolve ambiguities in matching at a lower level. The extent of a feature in an image is characterized by the highest and lowest epipolar line that the feature spans. The extents of matched features are identical in an ideal image because of the epipolar geometry.

At the top level, two bodies from the two structures with the same extents are considered as matching candidates. If a pair of surfaces from each body can be matched, then the body is labeled matched. The labeling can be revoked later if there is evidence that the initial match is faulty. Bodies with same extents are distinguished by the number of surfaces each body has, the order in which the surfaces are arranged, and the relative position of the bodies in each image. Any bodies that remained unmatched are matched again under less constrained requirements. Only one of the extents is required to be matched.

The system then attempts to match all surfaces belonging to each pair of matched bodies. The number of possible matches is only a fraction of the number of possible matches of all surfaces if they are segmented in bodies. Again the surfaces are labeled matched if they have the same extents. If more than one match satisfies the requirement, junctions belonging to the surfaces are matched. If that does not resolve the ambiguities, edgels belonging to the surface are matched. The best match is chosen from the pair of surfaces with the most number of matched pixels and fewest unmatched pixels. This, in effect, gives preference to surfaces with similar shape.

From a given pair of matched surfaces, we can easily match the curves and junctions since they are arranged in order. We build a three-dimensional depth map of the scene. Furthermore, the scene thus constructed is segmented and grouped together as surfaces and bodies. This facilitates surface interpolation and any further interpretation. Most other stereo systems give unsegmented results.

4 Examples

A pair of gray-scale images of some curved objects and blocks is shown in figure 2. The results from edge detection and edgel aggregation is shown in figure 3. It can be seen that a lot of edgels are missing around junctions. Other edgels are missing because of insufficient edge contrast. With a smaller directional operator, the curves are extended around the junctions. The results are shown in figure 4. The classification of junctions is shown. It can be seen that the classifier is not perfect. Some of the junctions humans would classify as T junctions are labeled as Y junctions. Segmented surfaces are shown in figure 5. The projection of recovered depth data from matched bodies and surfaces is shown in figure 6.

5 Constraint from Limbs

We pointed out in section 2.2 that limbs of a curved surface change with viewpoints. Limbs from different viewpoints correspond to different points on a curved surface. The curved surface is tangent to the lines of sight at the limbs. Within each epipolar plane, the curved surface is bounded by four tangent lines from two viewpoints. We can fit a planar conic curves to these four tangent lines. A combination of these fitted conic curves across all epipolar planes reconstruct the curved surface.

A conic has five parameters and we have only four constraints, thus one degree of freedom is available. Further constraint can be derived from shading, texture, symmetry, or others. Without these addition constraints, we choose the conic that has a least ratio of perimeter to area.

It can be shown that the equation of a conic tangents to four given lines is given by

$$L^2E^2 - 2L(AC + BD) + F^2 = 0,$$

where A, B, C, and D are the tangent lines; E, F the diagonals and

$$AC - BD = EF.$$

L is the fifth parameter which can be chosen freely. Some fitted conics to four tangent lines are shown in figure 7. The results from this part is not used in the example shown in section 4.

6 Conclusion

We have presented a hierarchical approach to stereo matching. This approach starts by building a hierarchical structures of features chosen for matching. Matching starts at the highest level of the structure. Results are used as constraints and guidance which are propagated to the lower levels. The resulting depth map is segmented and ready for surface interpolation.

This method gives globally consistent matching results and avoids local mismatches It reduces searching time for locating correspondence features during the matching process.

References

[1] D.Arnold, "Local context in matching edges for stereo vision," Proceeding: Image understanding workshop, May, 1978.

[2] Arnold, R. D., and Binford, T. O., "Geometric Constraints in Stereo Vision," Society Photo-Optical Instructment Engineers, Vol. 238, Image Processing for Missile Guidance, 1980.

[3] H.H. Baker, "Depth from edge and intensity based stereo," Ph.D. thesis, University of Illinois, September, 1981.

[4] T.O.Binford, "Inferring surfaces from images," Artificial Intelligence, Volume 17, 1981.

[5] S. Ganapathy, "Reconstruction of scenes containing polyhedra from stereo pair of views", Ph.D. thesis, Stanford University, 1976.

[6] D. Gennery, "Stereo camera calibration", Proceeding: Image understanding workshop, November, 1979.

[7] Gennery, D. B., "Modeling the environment of Exploring Vehicle by means of Stereo Vision," Ph.D. Thesis, Stanford University Computer Science Department Report STAN-CS-80-805, June 1980.

[8] W.E.L.Grimson, D.Marr, "A computer implementation of a theory of human stereo vision", Proceeding: Image understanding workshop, April, 1979.

[9] M.J. Hannah, "Computer matching of areas in stereo imagery," Ph.D. thesis, Stanford University, 1974.

[10] R.L. Henderson, W.J. Miller, C.B.Grosch, "Automatic stereo reconstruction of man-made targets", Society of photo-optical instrumentation engineer, Volume 186, Number 6, 1979.

[11] P. MacVicar-Whelan, T. Binford, "Curve finding with subpixel precision", Proceeding: Image understanding workshop, April, 1981. v

[12] J.Malik, T.O.Binford, "A theory of line drawing interpretation", Proceedings: Image understanding workshop, October, 1984.

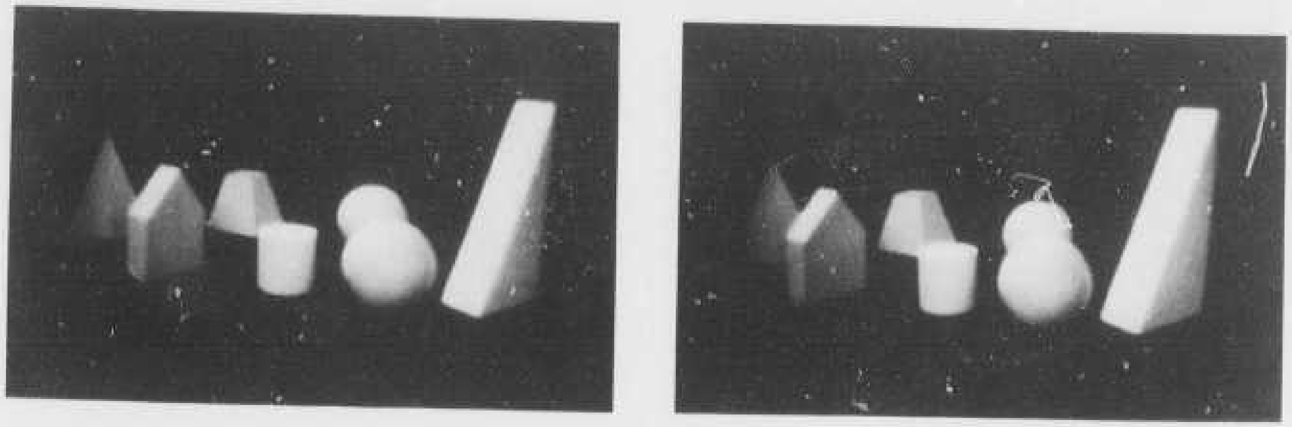


Figure 2: Gray Scale Images

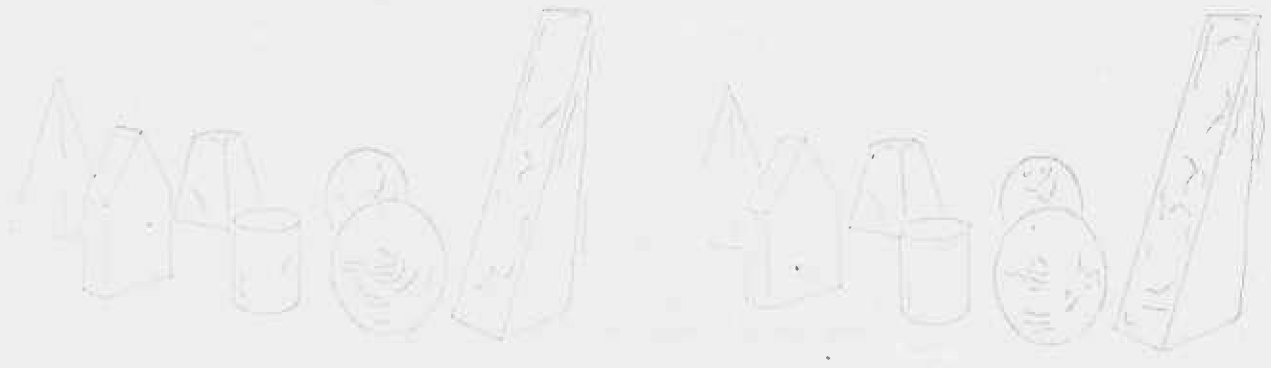


Figure 3: Detected Edgels with Curve Fitting

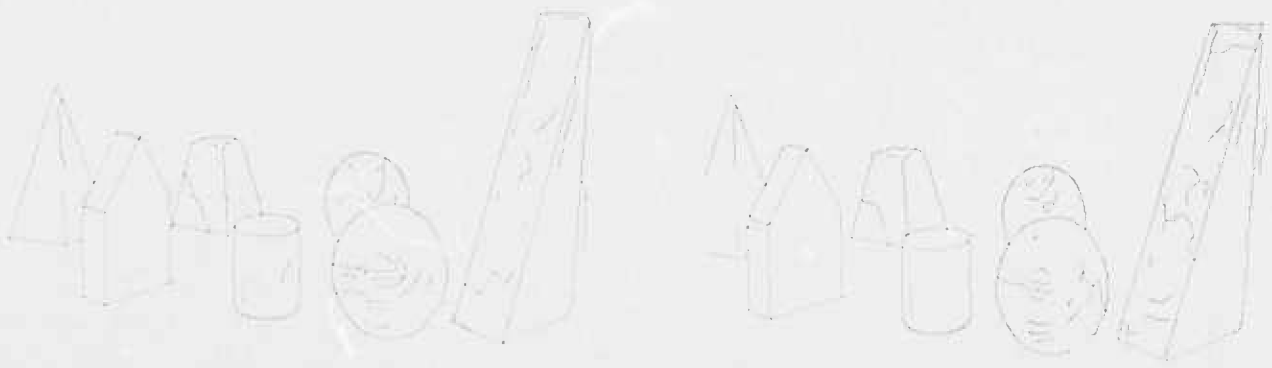


Figure 4: Results from Curves Extension

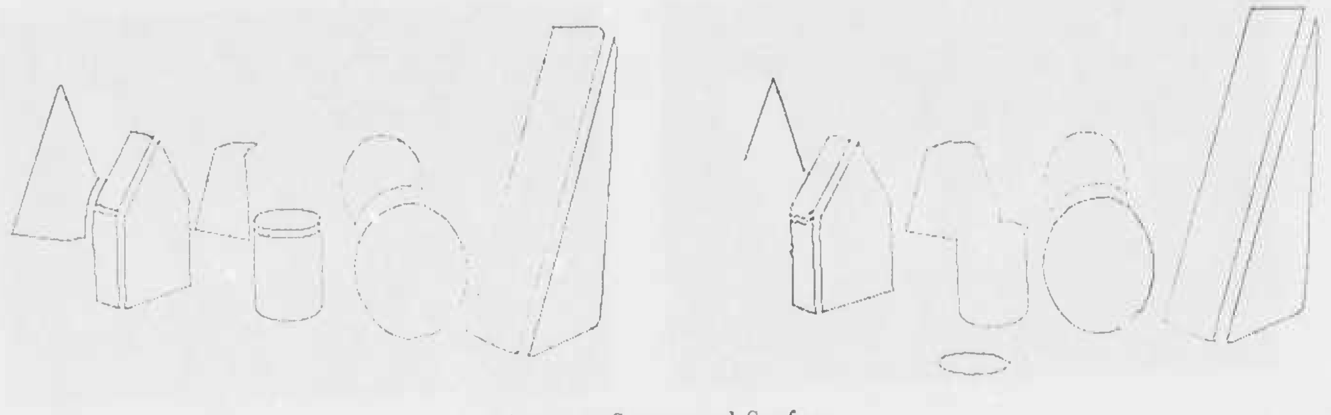


Figure 5: Segmented Surfaces

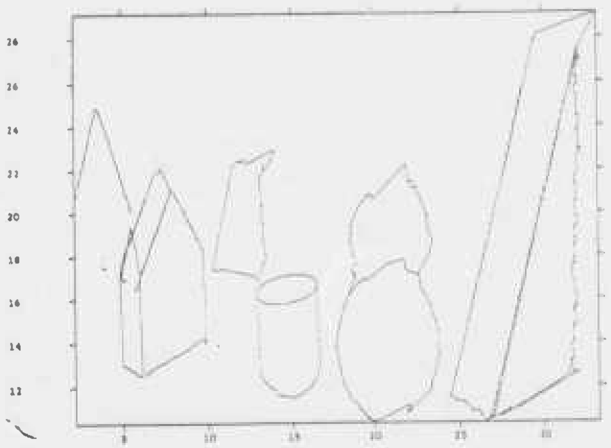


Figure 6: Projection of Recovered Depth Data from Matched Bodies and Surfaces

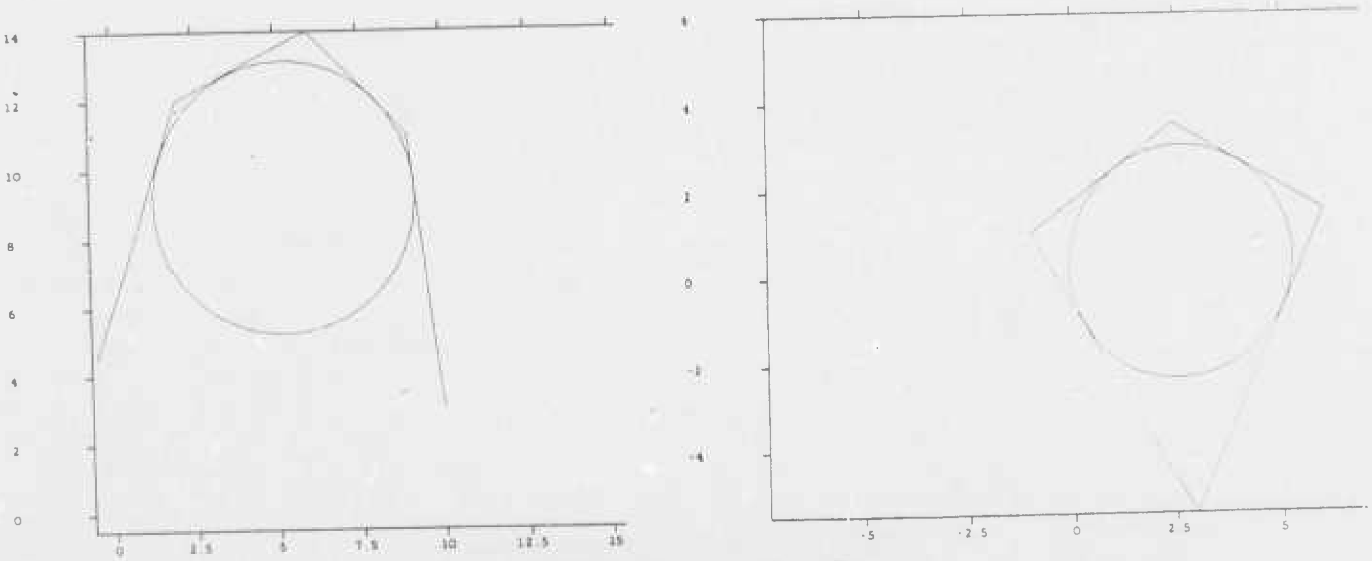


Figure 7: Fitting Conic to Four Tangents

[13] D.Marr, T.Poggio, "Cooperative computation of stereo disparity", Science, Volume 194, 1976.

[14] D.Marr, T.Poggio, "A theory of human stereo vision", MIT AI Memo, number 451, November, 1977.

[15] Meyhew, J. F. W. and Frisby, J. P. "Psychophysical and Computational Studies towards a Theory of Human Stereopsis," Artificial Intelligence 17, 1981. pp. 349-385

[16] G.M. Medioni, R. Nevatia "Segment-based stereo matching", Computer vision, graphics, and image processing, Volume 31, 1985.

[17] H.P. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover", Ph.D. thesis, Stanford University Computer Science Department Report STAN-CS-880-813, September, 1980.

[18] V. S. Nalwa, "On detecting edges", Proceedings: Image understanding workshop, October, 1984.

[19] V. S. Nalwa, E. Pauchon, "Algorithms for edgel-aggregation and edge-description", Proceeding: Image understanding workshop, December, 1985.

[20] Nishihara, H.K., Poggio, T., "Stereo Vision for Robotics," Proceeding of the international Symposium of Robotics Research, Bretton Woods, NH, September 1983.

[21] Y.Ohta, T.Kanade, "Stereo by intra and inter-scanline search using dynamic programming", Technical report CMU-CS-83-162, October, 1983.

[22] D. J. Panton "A Flexible Approach to Digital Stereo Mapping," Photogrammetric Engineering and Remote Sensing," Vol. 44, No. 12, pp. 1499-1512, 1978.

[23] D.N.Perkins, "Computer stereo vision: A combinatorial theory with implementation", Ph.D. thesis, Department of Mathematics, M.I.T., June 1970.

[24] J.Takamura, T.O. Binford, "Stereo modeling system: A geometric modeling system for modeling object instance and class", Proceedings: Image understanding workshop, October, 1984.

SYMBOLIC PIXEL LABELING FOR CURVILINEAR FEATURE DETECTION

John Canning, J. John Kim, Azriel Rosenfeld

Center for Automation Research, Computer Vision Laboratory
University of Maryland, College Park, MD 20742

ABSTRACT

A method for detecting linear features that are approximately one pixel wide has been developed. Each pixel has a set of possible masks associated with it, and the masks that are geometrically consistent with neighboring masks are identified. The consistency links between masks provide a means of constructing connected components and of extracting the mid-lines of the linear features that the components represent. The method is compared to the output from another line detection method. It is planned to use the new method as the bottom-up portion of a knowledge-based system for extracting extended linear features.

1. INTRODUCTION

Traditionally, curvilinear feature detection has been performed by algorithms that, given an input image, use a local operator to produce a numeric value for each pixel that indicates the likelihood that a line passes through that point (for examples, see [1,2,3]). Other algorithms generate line segments using mathematical operators such as Hough transformations. Still other algorithms find candidate starting points and sequentially follow the curvilinear feature until no more evidence can be found for its continuation (see [4,5,6]). These algorithms may have some control parameters such as thresholds that can be manipulated to improve the output to some extent. There are two problems with this approach when it is used as part of a scene analysis program:

(1) The scene analysis program controls the curvilinear feature extraction algorithm by means of numerical values and thus becomes an "algorithm expert" instead of a "curvilinear feature expert."

(2) The mathematical local operators provide no detailed justification for their decision to include or exclude a given point from the set of curvilinear features. At best, it may give some numeric measure of confidence in its decision, but this is only a summary of separate reasons for labeling the point as part of a curvilinear feature.

This research was supported by the Defense Mapping Agency under Contract DMA800-85-C-0007.

The system that we have constructed addresses these problems in the following ways:

(1) *Information preservation.* We try to keep all the individual sources of information, rather than replacing them by summary numeric measures, so that later stages can review the evidence used in making early decisions, even if the decision was to ignore the data. This also permits giving the user more detailed explanations of the chain of reasoning.

(2) *Symbolic instead of numeric reasoning.* We try to avoid numeric decision making (such as thresholding confidence measures) wherever possible. The input data, of course, is numeric so some of the reasoning will involve numeric comparisons, arithmetic, etc. Symbolic labels, however, are used at a very early stage to make the reasoning process more explicit.

Our goal is to build a system that can act like a human expert all the way down to the pixel level. In order to accomplish that, we feel that several principles must be followed.

(1) *Use all available knowledge.* Human experts justify their interpretations by using many types of knowledge. At the pixel level, determining that a curvilinear feature passes through a given pixel based solely on some measure of the contrast of the feature is not a good idea. If we are interested in curvilinear features of a specific type, such as roads in aerial photographs, we can use the knowledge of their local shape and appearance to improve the identification process.

(2) *Make minimal commitments.* Any system that locally identifies features is going to mislabel some of them because it lacks a global view of the situation. It is best to avoid commiting oneself to a unique labeling of a feature at the early (local) stages of identification so that alternative labelings can be examined in a more global view. This tendency towards least commitment must be balanced against the need to limit the combinatorial explosion that occurs when all combinations of labelings are considered globally.

(3) *Use order-independent methods.* When a method is order-independent, it can be more easily imple-

mented using parallel processing. Curvilinear feature following algorithms must operate sequentially and may produce different results depending on the starting point.

(4) *Justify decisions.* A human expert is able to explain why he/she identified a certain feature in a certain way. An expert system needs to emulate this behavior by recording all of its findings (information preservation) and by producing explanations of its decisions. This will also be useful in identifying the system's strengths and weaknesses.

The problem domain for our curvilinear feature detection system is road networks in aerial photographs. We have limited ourselves to trying to detect roads that are approximately one pixel thick, so that we can detect their presence at the 3×3 neighborhood level (see [7]). In actuality, if we view the gray levels of the image as the third (z-axis) dimension of the image, our system finds "ridges" (or "valleys"). The thickness of a curvilinear feature is related to, but not completely dependent on, the sharpness of the peak (or valley); thus as long as a curvilinear feature wider than one pixel has a ridge (or valley), we can detect it. We further restricted our problem domain by not allowing roads to have high curvature, such as hairpin turns.

The output of our system is a set of curvilinear feature fragments for an expert system to reason about. The curvilinear feature fragments are generated using the following four stage algorithm. For each pixel p we generate a set of masks that describe the pattern of gray levels in the 3×3 neighborhood centered at p (see [7]). The set of masks is shown in Appendix A. Considering each mask to be a hypothesis about p, we want to construct consistent interpretations from the hypotheses. We identify pairs of neighboring masks as "consistent" as described in Section 2. Two consistent masks are said to have a consistency link between them, and the transitive closure of these consistency links is used to build connected components representing consistent hypotheses. The next stage of the system finds the midlines of the connected components, as described in Section 3, so that the curvilinear features can be represented as simple chain codes. Section 4 deals with the splitting of the curvilinear features into pieces having exactly two endpoints. Section 5 presents the results of applying the system to an image, and Section 6 compares the results with the output of a standard non-linear line detection algorithm.

2. MASK CONSISTENCY

The mask matching stage (described in detail in [7]) finds all "interesting" 3×3 masks that "match" the neighborhood of each pixel. Here we are using local shape information to identify the pixels that might have curvilinear features passing through them. Our principle of minimal commitment leads us to build a system which labels each pixel with several possible patterns instead of just one based on some numerical measure.

A mask is said to "match" a given 3×3 neighborhood centered at a pixel if there exists a way of classifying the pixels of the neighborhood into two groups, foreground and background, that form the same pattern as the binary mask. For a monochrome image, the classification of pixels into two groups can be performed by thresholding the gray level values. The nine values in the neighborhood naturally define ten intervals (fewer if some of the pixels have the same gray level) at which such a threshold could be placed. Eight of these intervals yield binary patterns containing both foreground and background pixels. Our system checks each of these intervals to see if thresholding in that interval yields an "interesting" pattern (i.e. one that matches one of the masks in Appendix A). All such matches are recorded for each pixel along with the width of the interval over which the threshold can range. We call the width of the interval the *robustness* of the mask because it measures the separation between the foreground and the background in gray levels. The larger this separation, the more resistant is the difference between foreground and background to being altered by noise, so that the corresponding threshold is more "robust."

Our "matching" criterion is the weakest condition that a human expert would use when faced with the same task. Figure 1 shows a sample 3×3 gray level neighborhood on the left. When looking for the possibility of a bright line passing through the central pixel of that neighborhood, an expert would conclude that there is evidence for such a line only if he/she found that the brightest pixels in the neighborhood were arranged in a line-like formation. Our set of masks (Appendix A) defines these line-like formations and our matching criterion guarantees that the line pixels are brighter than the non-line pixels. Figure 1 also shows all the masks that match the sample 3×3 neighborhood. The gray level values, in order, are 66, 79, 89, 91, 91, 94, 104, 133, and 150. The non-trivial threshold intervals are [66, 79), [79, 89), [89, 91), [91, 94), [94, 104), [104, 133), and [133, 150). The robustness value is given below each mask where it is defined (the uniform masks don't have a separate foreground and background). Thus we have a record of all the possible labels that describe the gray level pattern at the given pixel. The labels are symbolic descriptions that will be the lowest level in the chain of reasoning.

In order to build more global interpretations of the data output from the mask matching stage, we need to determine when masks (hypotheses about the surrounding pattern) are consistent. Noting that the masks centered at two adjacent pixels (in the eight-connected sense of adjacency) correspond to two overlapping 3×3 neighborhoods, we define mask consistency in the following way:

(1) *Joint robustness greater than 0.* There must exist at least one gray level that can serve as a threshold value for both masks. This means that there is a non-zero overlap in the ranges of threshold values (robustnesses) for the two masks.

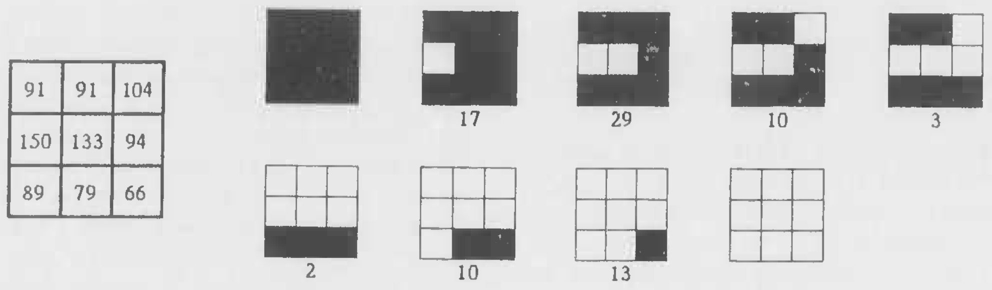


Figure 1. Masks and robustnesses for a sample 3$\times$3 neighborhood

(2) *Identical overlap.* The two binary masks must have the same black and white pattern in the region where the two 3$\times$3 neighborhoods overlap.

(3) *Geometric consistency.* The two masks must make geometric sense in the given problem domain. This means that no domain assumptions are contradicted by the combined mask pair.

The first two conditions guarantee that the gray level values of the two masks are compatible. The third condition guarantees that the two masks make geometric sense in the problem domain.

Two consistent masks may also suggest a new labeling for the two masks that was not in the original set of labels. For instance, two anti-parallel edge masks when placed together suggest that a two-pixel thick line passes between the two center pixels. The new labeling is then added to the set of labels for the two masks. This is a trivial example of a top down action where a slightly more global view has suggested a change in a lower level of the system.

In our problem domain, there are two different types of consistency between masks. The line joining the centers of the two masks may be parallel to the direction of a road or it may be perpendicular to it. We have divided the consistency types into two classes, "extending" and "broadening." Figure 2a shows that extending consistency occurs when the line joining the two mask centers is within $\pm 45^\circ$ (inclusive) of the direction of the curvilinear feature. Broadening consistency occurs for the other orientations. Of course, the direction is not very well defined when we are given only two overlapping 3$\times$3 neighborhoods, so the classification of the consistency type between each pair of masks was done subjectively. The strength of geometric consistency varies between pairs of masks meeting the three consistency criteria, so we created two weak consistency types, "maybe-extending" and "maybe-broadening," to capture that notion. Further refinement of the strength scale into more than two classes is possible but does not seem necessary. We will refer hereafter to "extending" and "maybe-extending" ("broadening" and "maybe-broadening") as consistencies of extending (broadening) type. Figure 2b shows two examples of each of the four consistency types. Appendix B gives the complete list all possible overlapping masks and their assigned consistency types. The short names written under each pair of masks in the appendix are mnemonic names used to further classify the consistency types during the labeling process; they were not used in the algorithm.

We now have a symbolic description of the geometric relations between the pixels that might have curvilinear features passing through them. No commitment has been made as to what is the best description, but we have noted which descriptions are consistent and used slightly more global shape information. Note that the two stages (mask matching followed by consistent pair labeling) can each be implemented as pixel parallel operations.

3. MID-LINE DETECTION

In the third stage, we connect the local patterns found in the first stage using the consistency criteria of the second stage to form more global hypotheses about curvilinear features in the image. We now can use the geometric knowledge recorded in the consistency links to

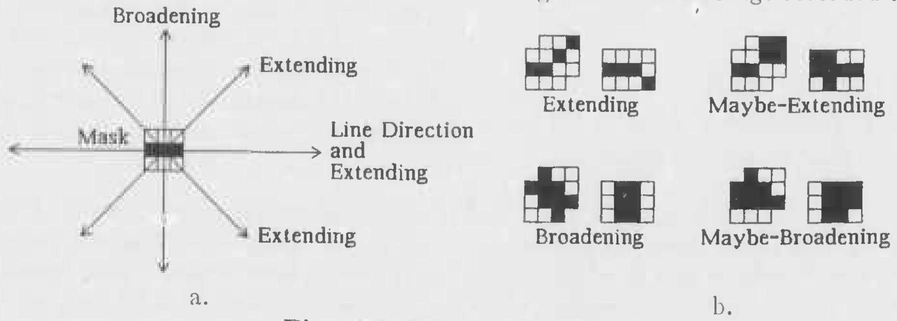


Figure 2. Consistency types

determine where the mid-lines of these curvilinear features lie.

Connected components are derived from the transitive closure of the consistency links. We used the algorithm presented in [8] (adapted to our data structures) to label the connected components. Each connected component is given a numeric label for later reference.

The expert system that will use the output of the curvilinear feature detector will need to know the mid-line of each connected component in order to reason about the geometry of the road network. Thus, if a connected component is "thick," we must decide which masks are on the mid-line and which are not (a "thin" connected component can be the same as its mid-line). A component is "thick" if it contains a two by two block of pixels that are all in the component, as shown in Figure 3, because we can no longer represent it as a list of

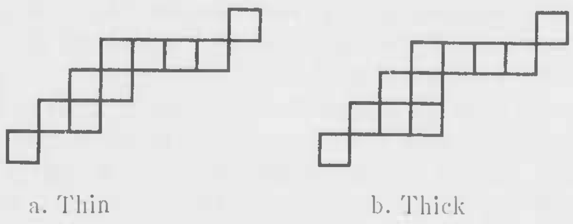


Figure 3. Thick and thin connected components

8-adjacent pixels where the orientation of the line segments between consecutive pixels is parallel ($\pm 45^\circ$) to the local orientation of the curvilinear feature.

Mid-line detection is similar to the standard image processing operations of thinning and skeletonization. It differs from those operations in that we label some of the pixels as being "not on the mid-line" but do not discard them, and we label the pixels by using the semantics of the consistency links instead of the shape of the connected component. A set of heuristics was used to progressively label those pixels not on the mid-line as "suppressed." The consistency links are marked with suppression labels at their endpoints and a mask is fully suppressed when all of the endpoints of consistency links attached to it are suppressed. The heuristics fall into three groups:

B The "B" heuristics look for broadening links and suppress the endpoint not on the mid-line or both endpoints if the link is redundant (i.e. there are extending links attached to masks B and F in Figure 4 that are perpendicular to the line BF). The mask at the endpoint having the most extending links attached to it is considered to be closest to the mid-line.

E The "E" heuristics decide which of two parallel extending links is not on the mid-line and suppress both endpoints of that link (see Figure 5). Again, the link having the most extending links attached to the masks at its endpoints is viewed as the link being closest to the mid-line.

D The "D" heuristics clean up the consistency links that were not suppresed by the B and E heuristics but do not lie on the mid-line. The first looks for masks that are "dangling" from the mid-line by one unsuppressed broadening link. The second is applied after the first and arbitrarily decides which masks must be suppressed to achieve a thin midline if all the other heuristics have failed.

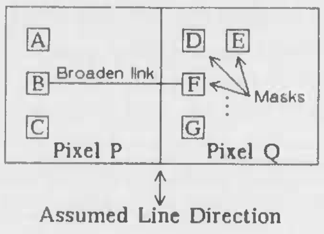


Figure 4. Suppression of broadening type consistency links

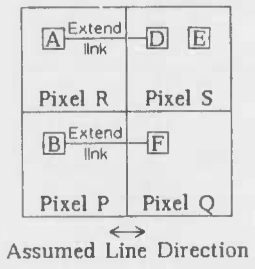


Figure 5. Suppression of extending type consistency links

See Appendix C for full details on the heuristics. If suppression of a mask would break a component into two or more smaller components (using pixel-wise connectivity), no suppression occurs. The first D heuristic, however, will completely erase connected components having only two masks connected by a broadening link. Some examples of the applica on of these rules can be found in Section 5.

To see how much each rule helped in determining the mid-lines, we calculated some statisitics on their frequency of occurence for one image (the image presented in Section 5). Table 1 shows the percentage of suppressed masks that were suppressed for a unique reason. Masks that were suppressed for multiple reasons are analyzed in Table 2 which gives the breakdown of suppression rules summing over all such masks. Table 1 shows that

- Masks are rarely suppressed only because they lie on parallel extending links.
- Rule D1 is being used a bit more frequently than is desirable, but its use may be due to the large number of components containing only two masks joined by a broadening link.

Table 2 shows that

- When we count the reasons for suppressing individual links instead of masks, the more "knowledgeable" rules are being used most.

Labeling masks as suppressed is consistent with our philosophy of making minimal commitments and preserving information; we don't commit ourselves by throwing away the pixels that are not on the mid-line. When we later go back to check the justification for our decisions in this stage, we can look at which heuristics were used in the suppression of each mask. Some of the decisions were based on weighted sums, but those sums are really counts of the number of times different kinds of links occured. Thus, we used the symbolic knowledge

Suppression of masks for unique reasons	
Suppression rule	Percentage
B1	16.2%
B2	9.35%
B3	13.4%
E1	4.90%
E2	0.202%
E3	0.076%
D1	21.7%
D2	3.87%
Mixed reasons	30.2%

Table 1.

Suppression of links in masks suppressed for mixed reasons	
Suppression rule	Percentage
B1	34.0%
B2	12.3%
B3	4.83%
E1	30.8%
E2	4.21%
E3	0.72%
D1	6.32%
D2	6.82%

Table 2.

generated in the previous stage as well as general shape knowledge to perform the suppression. All of the suppression heuristics except for D2 are pixel-parallel operations (but the D rules must be performed after the B and E rules are applied, and D2 must be applied only after all the others).

In future work, we may incorporate the knowledge stored in the gray level image to localize the mid-lines under the assumption that the mid-line is the brightest/darkest part of the linear feature. The techniques of sub-pixel localization could then be applied.

4. SPLITTING COMPONENTS INTO SEGMENTS

The mid-lines of the connected components can form complicated graph structures. The reasoning system that will construct the road network from the curvilinear feature fragments is more easily implemented if all of the fragments have exactly two endpoints (because all of the endpoints must be investigated and having a constant number of endpoints simplifies the enumeration). All graphs can be broken down (by eliminating links) into a set of subgraphs each of which:

1) shares no nodes with any other subgraph,

and either

2a) has two nodes of degree 1 and n nodes ($n \geq 0$) of degree 2,

or

2b) is a cycle consisting entirely of nodes of degree 2,

or

2c) is an isolated node.

We call each such subgraph a *segment*. Note that the mid-line is viewed not as a series of links between masks, but as a series of links between pixels containing at least one mask belonging to the connected component. As a consequence of this viewpoint, the mid-lines of two connected components may pass through the same pixel (since the pixel can contain masks belonging to different connected components).

We split the mid-line in two stages in order to have a set of *segments* and *loops* that have two and no endpoints, respectively (a segment may be of length 1 in which case its two endpoints are the same point). Loops in the mid-line are rare in our problem domain but do occur. In terms of road structures they can occur at the 270° turn of a highway cloverleaf or at the dividing and remerging of a divided road. One could choose to view a loop as a segment whose endpoints are adjacent, but we found that it was useful to explicitly label them as different from segments because they occur more frequently on road structures. The loops are extracted from the mid-line first, and then any remaining points are split into segments. We still preserve, however, the information concerning the original structure of the mid-line so that we don't commit ourselves to a decision based only on the local properties of the mid-line.

Loop and segment extraction use the outer border of the mid-line (as defined in [9]) to find the respective structures. To find the outer border, we use a variation of the border following algorithm given in Section 11.2.2b of [9] that chooses 4-adjacent neighbors for the next point on the perimeter whenever possible (see, for example, the staircase edge at the left of Figure 6). The first border point is chosen such that no other point in the mid-line is on a higher row (to guarantee that this is the outer border). Let S be the set of points in the mid-line and let B be the outer border of S. Loops can be detected by finding a point q in B that has two distinct neighbors in B, but is only visited once during a counterclockwise walk around the border. Once point q is detected, any point that appears more than once in the counterclockwise walk around the border, starting at q, is a branch to a segment or loop not on the loop containing q. Such branches are not included in the loop. When a loop is found, its points are removed from B, and the remaining points are searched for more loops until none can be found. The points that remain in B

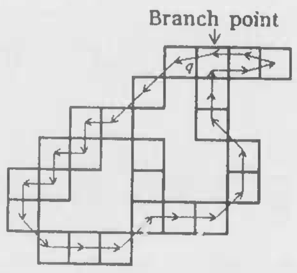


Figure 6. Counterclockwise walk around the outer border of a mid-line

after all the loops have been removed are then searched for segments. Segments are made up of points in B that have two distinct neighbors in B, and start and end at points having other than two neighbors in B. As each segment is discovered, its points are removed from B until no more segments can be found (i.e. B is empty). The process of loop and segment extraction is repeated by creating a new set B' for the outer border of $S - B$ until B' is empty.

The segments and loops extracted from each mid-line are numbered to identify them. Points in each connected component that were not on the mid-line (i.e. the pixels that were suppressed in the previous stage) are then reattached to the "closest" segment or loop. "Closeness" of a pixel p is determined using the consistency links from all of the masks belonging to the given connected component. The system reattaches p to the segment connected to p by the strongest consistency link, where the different kinds of consistency links are ranked (from strongest to weakest): broadening, maybe-broadening, maybe-extending, extending. This ranking reattaches pixels by looking in the direction perpendicular to the curvilinear feature direction first and then in the other directions. In the case of two consistency links of the same type, the one with the larger joint robustnes (see Section 2) is stronger. If the two links are still of equal strength, the tie is broken arbitrarily. If a pixel does not have a consistency link to another pixel on the mid-line, then it must be connected through a chain of consistency links having length greater than one. After reattaching all masks with direct consistency links to a mid-line, we iterate the procedure, attaching other pixels to the newly labeled pixels, until all the pixels not on the mid-line are reattached.

This final stage of the algorithm has again used the information garnered from the previous stages (along with knowledge about the topology of graphs) to make its decisions. We preserve the information from the previous stages and add to it a set of decisions about how the mid-lines should be broken up to make segments. This algorithm can be implemented in parallel for each connected component (but not for each pixel).

5. RESULTS

Figure 7 shows a 100×100 gray level image digitized from an aerial photograph of the Greenbelt, Maryland area. The image is a sub-sampled version of a 200×200 pixel region of the original digitized image. The width of the roads in the image varies from less than 1 pixel to almost 3 pixels. Three major roadways are visible (along with two partial cloverleaf intersections) and some suburban roads. A shopping center appears in the lower right quadrant of the image.

The masks found for a small section of the image represented by the square in Figure 7 are shown in Figure 8. Only bright lines on dark backgrounds were

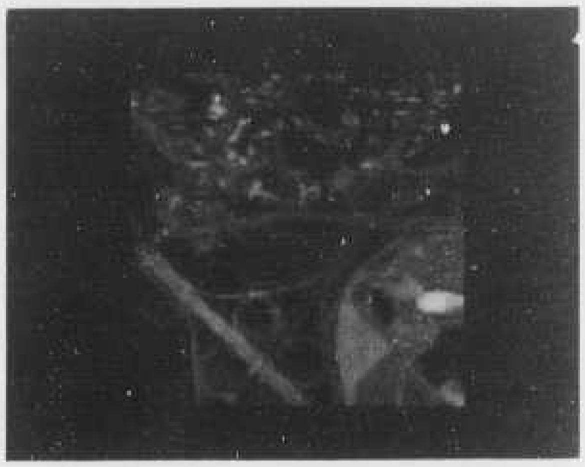


Figure 7. Input gray level image

searched for. The numbers underneath the masks indicate the robustness of each mask (i.e. the range of gray level values that can be used as a threshold to separate the bright pixels from the dark pixels in the 3×3 neighborhood). Note that the masks having the largest robustness for the horizontal road near the top of the region represent fairly well the pattern of the road. This is not true in general, however, as was found in [7].

Figure 9 shows the consistency links for the same region. The numbers beneath the masks now indicate the connected component number to which the mask belongs. Masks that have no number were not consistent with other masks. Some of the border masks are consistent with masks not in the region. Broadening type consistency links are shown as dashed lines while extending type consistency links are shown as solid lines. Some of the links in the figure overlap making it more difficult to determine the masks to which they are connected.

Figure 10 shows the result of mid-line detection and segment splitting. Masks not on the mid-line have not been drawn. The numbers beneath the masks here indicate the segment number to which the mask belongs. Note that some of the connected components generated no segments because they had no extending type relationships (e.g. component #483 near the lower right

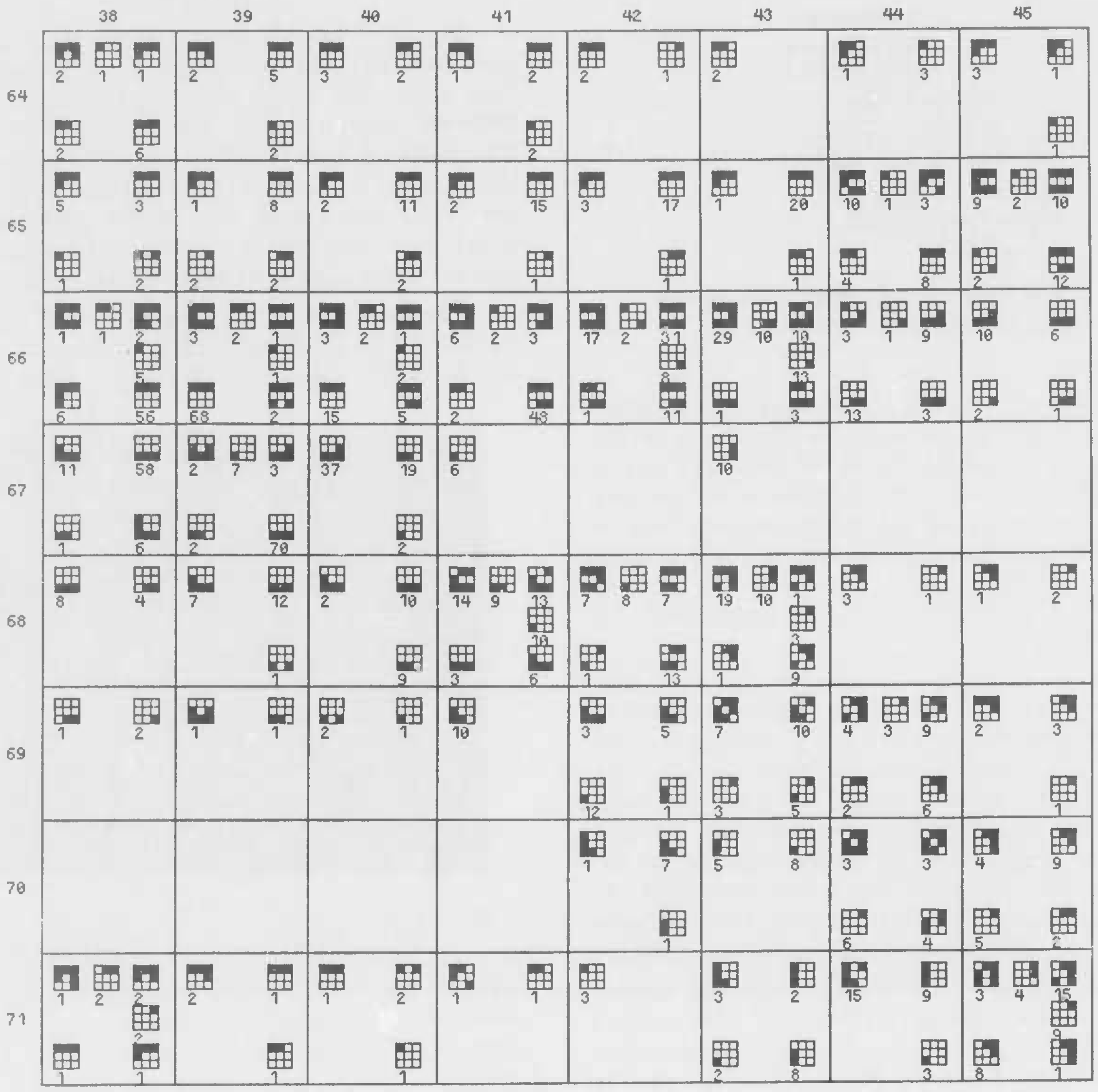


Figure 8. Masks found for the small square in Figure 7

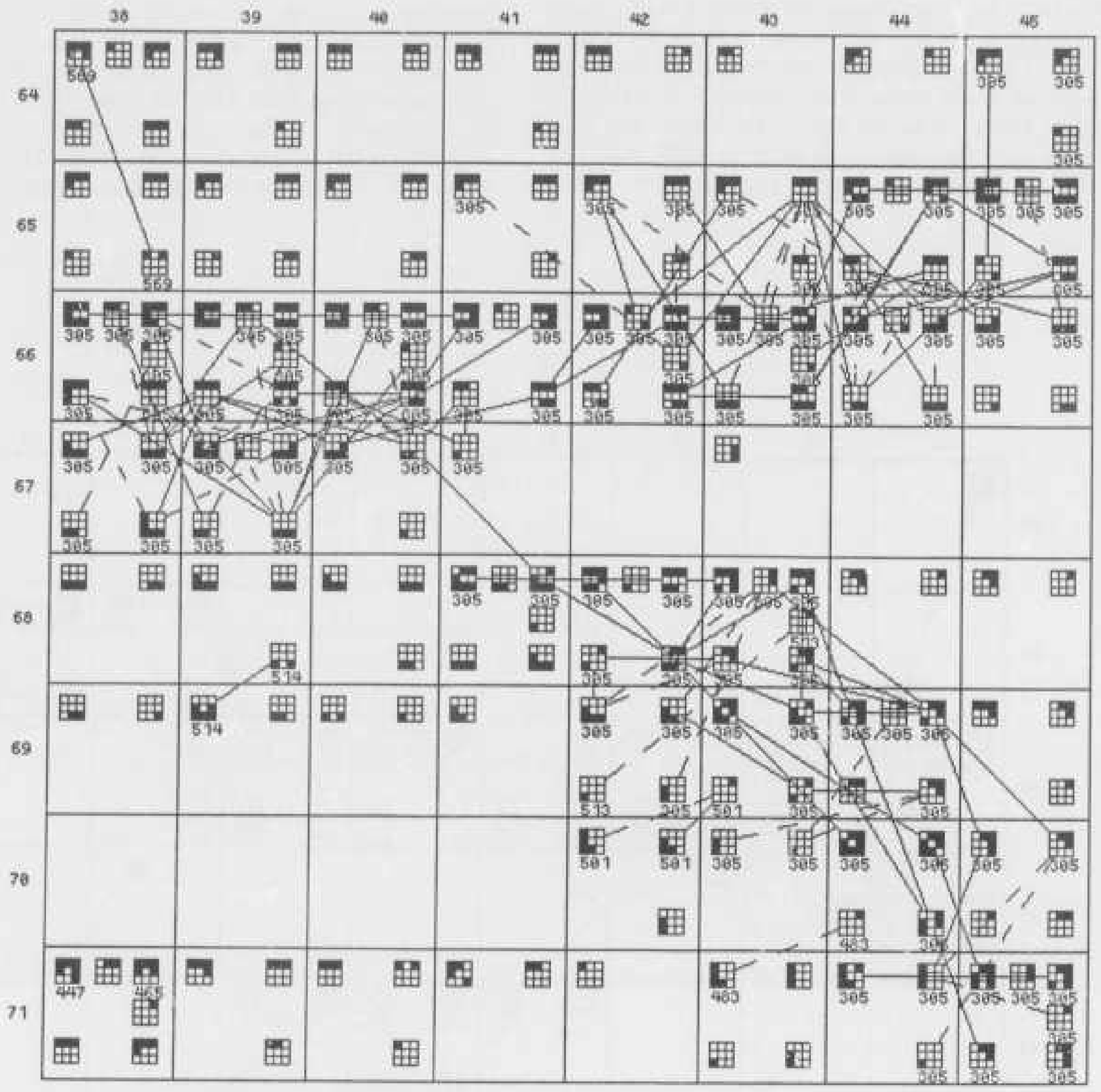


Figure 9. Consistency links for the small square in Figure 7

hand corner). The largest segment, #233, is really part of a loop that goes beyond the borders of the 8×8 region shown here. The loop (#233) is still "thick" since it has a 2×2 block of pixels containing unsuppressed masks in the extreme lower right corner. The reason for this thickening is that the edge mask at pixel 45,75 still has unsuppressed maybe-extending consisistency links attached to it and the results of the final suppression heuristic (rule D2) are not shown in this figure. A possible heuristic to handle such cases would be to suppress maybe-extending links that lie ±45° from an extending link belonging to the same component and sharing a common pixel at one endpoint. Note that component #305 was broken into loop #233 and segment #229.

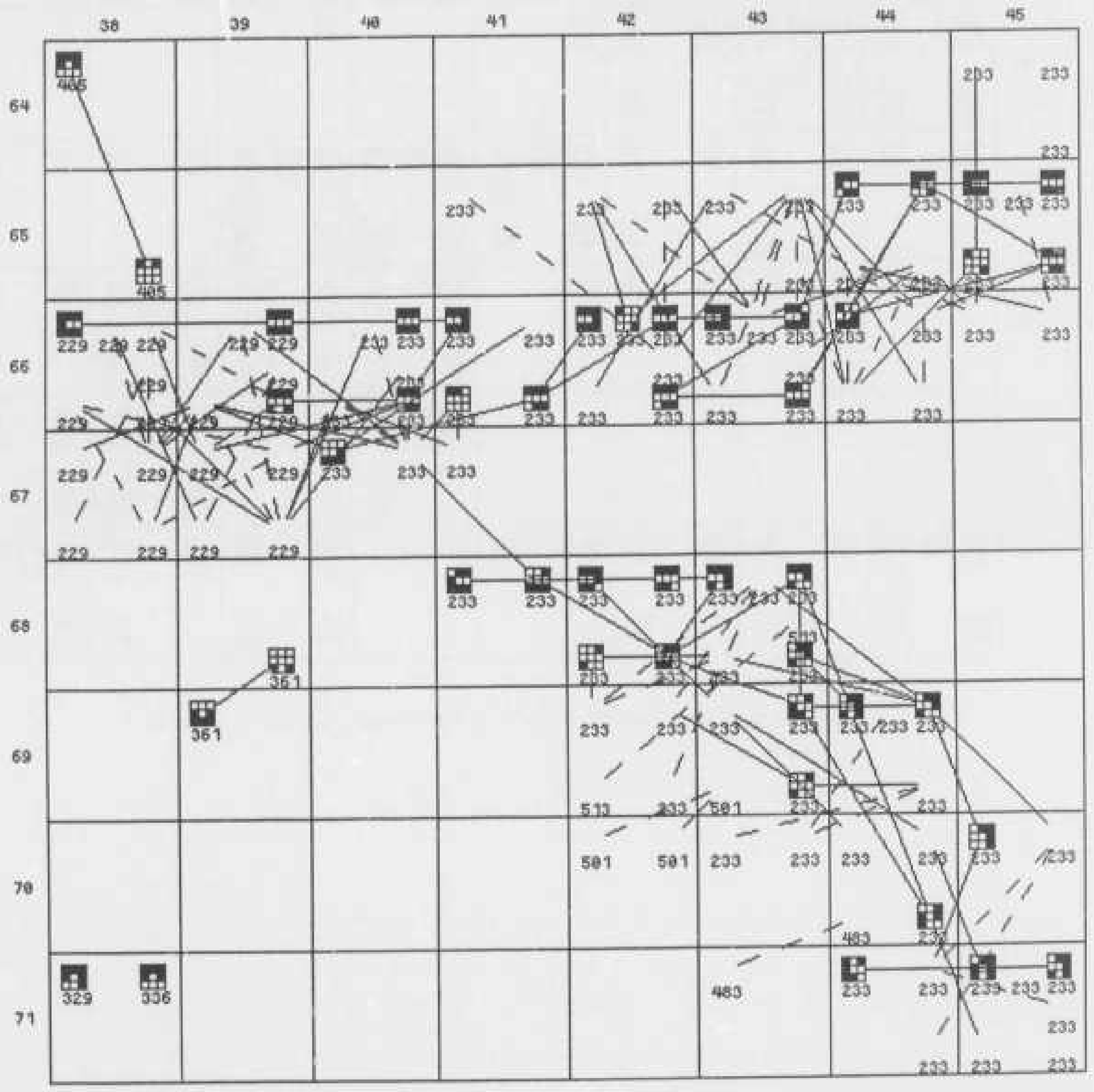


Figure 10. Mid-lines of the segments in the small square of Figure 7

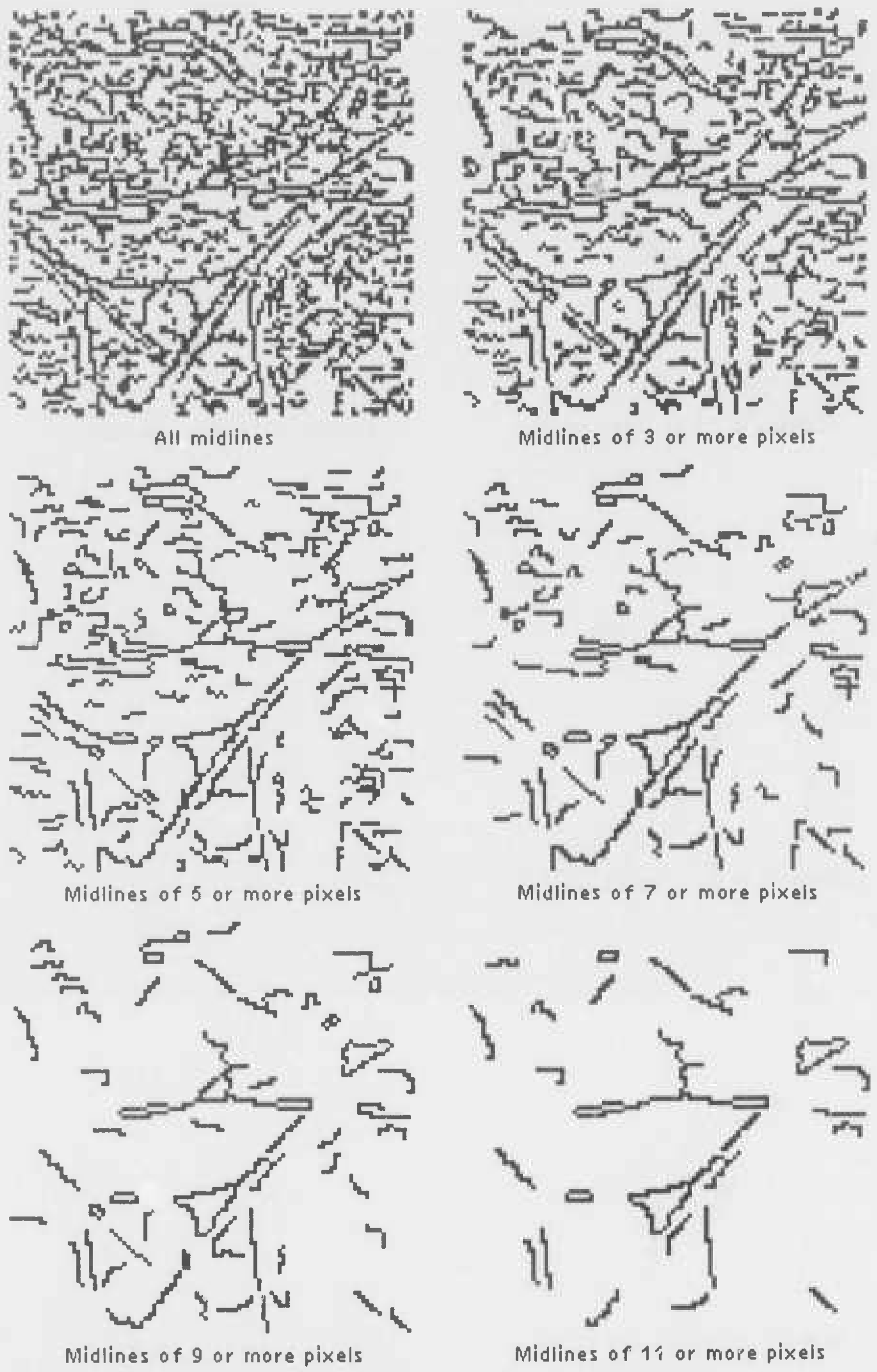

All midlines
Midlines of 3 or more pixels
Midlines of 5 or more pixels
Midlines of 7 or more pixels
Midlines of 9 or more pixels
Midlines of 11 or more pixels

Figure 11a. Bright mid-lines of the segments found in Figure 7

Figure 11(a and b) shows all of the mid-lines of all of the bright segments on dark backgrounds found in the input image. In order to show the lengths of the mid-lines, the individual frames of the figure show mid-lines containing more pixels than a given threshold. When interpreting this figure, keep in mind that two mid-lines can pass through the same pixel and that two mid-lines passing through adjacent pixels may look like one long mid-line or a thickening of a mid-line. The original 1742 connected components found in the image in Figure 7 yielded 1231 segments and loops.

In general, the longest segments correspond to identifiable roads in the image, but there are many short, false-alarm segments. Loops tend to occur on divided highways and cloverleaf interchanges and are either elongated with two parallel sides or have a polygonal shape with uniformly curving sides. The loops that are not on roads tend not to have long, straight edges. The suburban streets do not show up well, because the areas immediately surrounding them are not uniformly darker than the road (the roadsides are dotted with man-made structures which appear brighter than vegetation or earth). Searching the image for dark segments on bright backgrounds does not bring them out very well either since the background is both brighter and darker than the street itself.

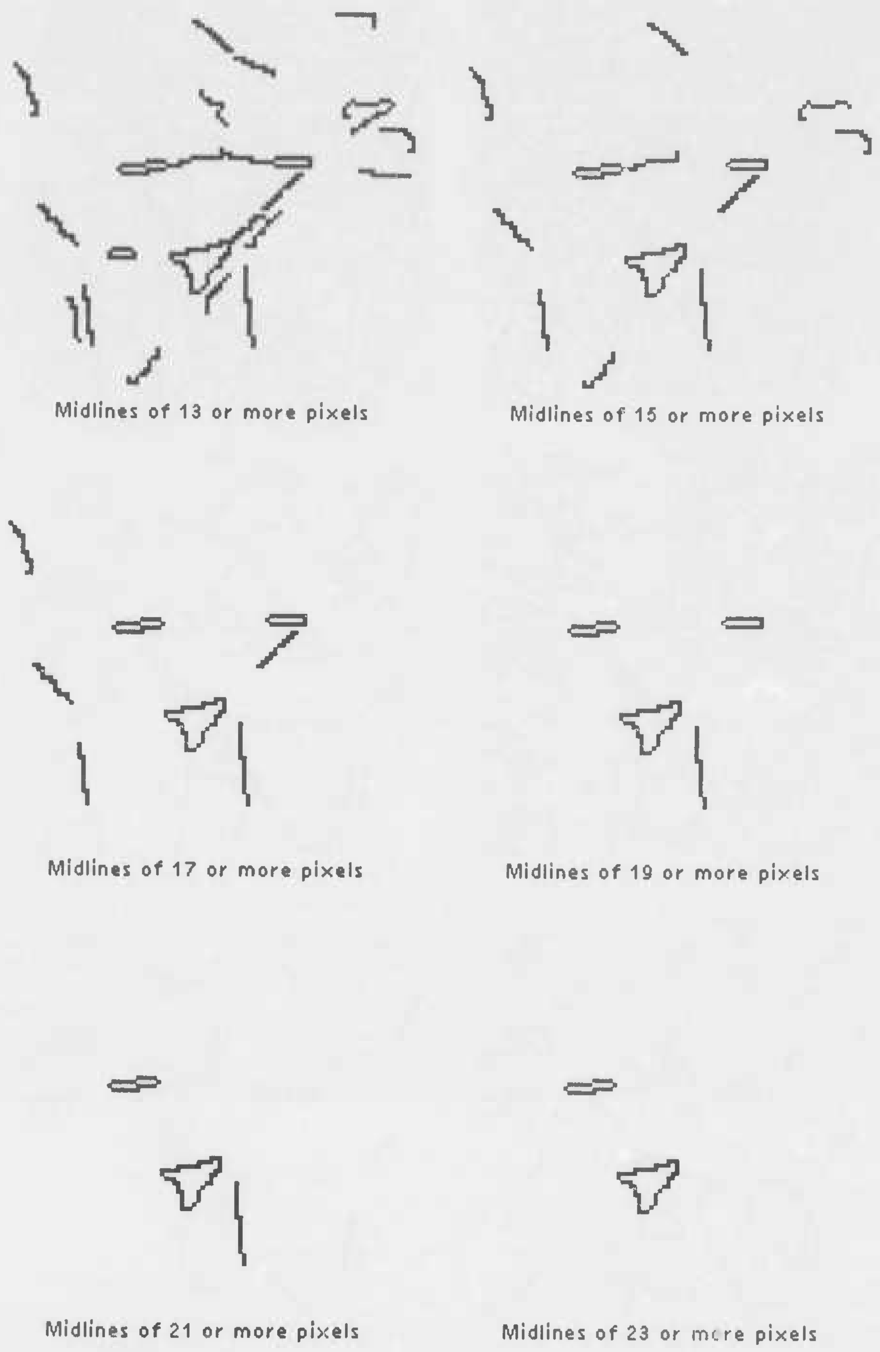
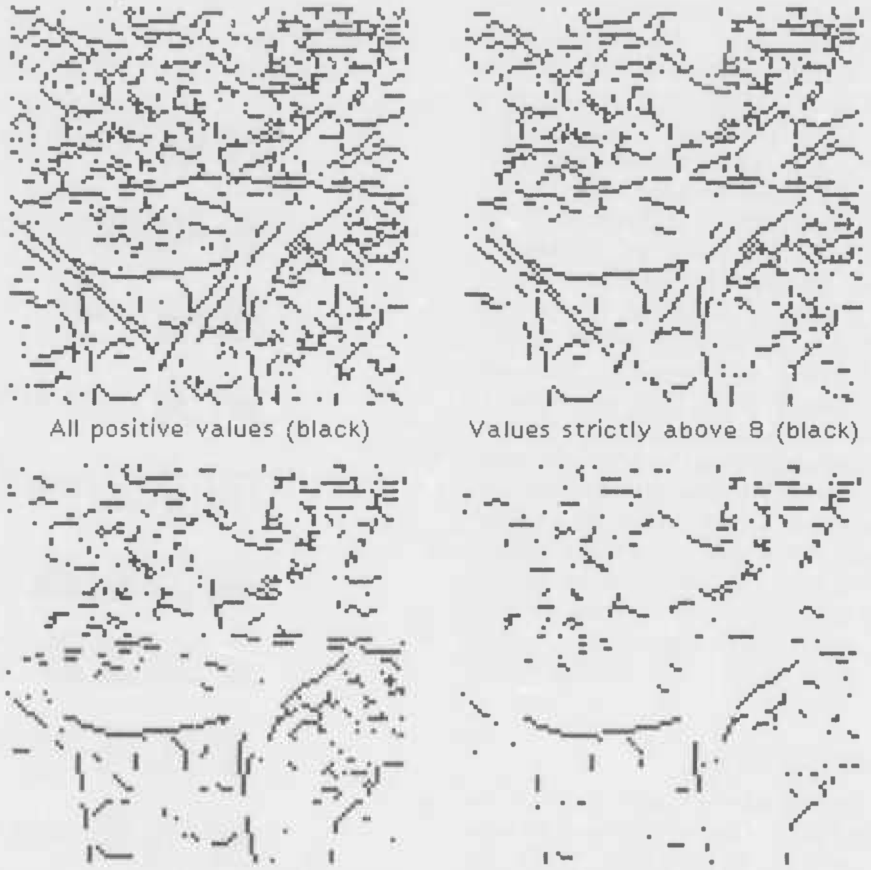

Midlines of 13 or more pixels
Midlines of 15 or more pixels
Midlines of 17 or more pixels
Midlines of 19 or more pixels
Midlines of 21 or more pixels
Midlines of 23 or more pixels

Figure 11b. More bright mid-lines of the segments found in Figure 7

The segments were derived using local shape information only. Gray level values have not entered into the decision criteria other than in our requiring masks to have positive robustness and joint robustness. Contrast and absolute gray level information has been recorded in the data structure for further examination, but has not been used to eliminate segments from consideration. This preservation of information may lead to many false alarms in the detection process, but it provides a much richer source of information for the reasoning system to work on. Other information for the reasoning system, such as the orientation of the endpoints of the segments, the average color (gray value) of the segments, and the segment numbers of nearby segments is stored for each segment, but is not shown here.


All positive values (black)

Values strictly above 8 (black)

Values strictly above 16 (black)

Values strictly above 24 (black)

Figure 12. Non-linear line detector output from Figure 7

6. COMPARISON WITH A NON-LINEAR LINE DETECTOR

Figure 12 shows the results of the non-linear line detector described in Section 10.3.2 of [9] applied to the image in Figure 7. The magnitude of the operator is the average difference between the three pixels lying on the "line" and their immediately adjacent neighbors perpendicular to the line. The operator's direction output is not shown here, but it gives the orientation of each of the two endpoints of the line segment passing through the pixel. Comparing the points having positive output in the non-linear line detector with the mid-lines of the consistency link algorithm before length thresholding, we can see that both methods detect the cloverleaf intersection in the lower left, as well as most of the major roads. The non-linear line detector's results are a bit more fragmented, especially along the divided highway between the large cloverleaf and the smaller one, but neither method shows the roads clearly while excluding the non-road areas.

The big difference between the two methods, however, is not in the appearance of their outputs, but in their information content.

- Our segments contain width information (broadening type consistencies) not present in the non-linear line detector outputs.
- Multiple hypotheses about the pattern of gray levels at each pixel are retained in our algorithm.
- Curvilinear feature junctions are not explictly represented in the non-linear line detector output.
- Two-pixel thick lines can be detected by our method, but may be missed by the non-linear detector.
- In our system consistency between neighboring pixels in a curvilinear feature is constrained by "semantic" rules (concerning curvilinear features corresponding to roads) while the non-linear detector doesn't specify how to determine the consistency.
- Our system can justify its decisions by giving reasons for the labeling of each stage whereas the non-linear detector must base its decisions only on the magnitude of its contrast measure.

7. CONCLUSIONS

A method for detecting curvilinear features using symbolic consistency links has been developed. The system *preserves information* instead of making simple thresholding decisions to filter out noise and *reasons symbolically* about the shape of the curvilinear features it is trying to detect. Each pixel can have multiple hypotheses about its role(s) in curvilinear features passing through it so that we make *minimal commitments* about its final labeling. Consistent hypotheses about the local patterns of gray levels are connected together to form larger hypotheses. The resulting hypotheses are in the form of segments (chains of consistent masks) that can be used by a higher level reasoning system to find road networks. The chain of reasoning for each hypothesis is explicitly stored in the data structures to *justify decisions*. *Order-independent methods* were used to permit parallel implementation and to avoid the problem of having the final output depend on the choice of starting points. Each stage makes use of *all available knowledge* such as local curvilinear feature appearance, geometric consistency of local patterns in the domain of roads, and graph topology. The new method provides more information for the reasoning system and performs as well as or better than simple line detection algorithms.

REFERENCES

[1] Canny, J., "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 6, pp. 679-698, November 1986

[2] Kim, J. J., "Road Detection on Radar Imagery," Masters thesis, Virginia Polytechnic Institute, July 1985

[3] Vanderbrug, G. J., "Line Detection in Satellite Imagery," *IEEE Transactions on Geoscience Electronics*, Vol. GE-14, No. 1, pp. 37-44 January 1976

[4] Fischler, M. A., Tenenbaum, J. M., and Wolf, H. C., "Detection of Roads and Linear Structures in Low-Resolution Aerial Imagery Using a Multisource Knowledge Integration Technique," *Computer Graphics and Image Processing*, Vol. 15, pp. 201-223, 1981

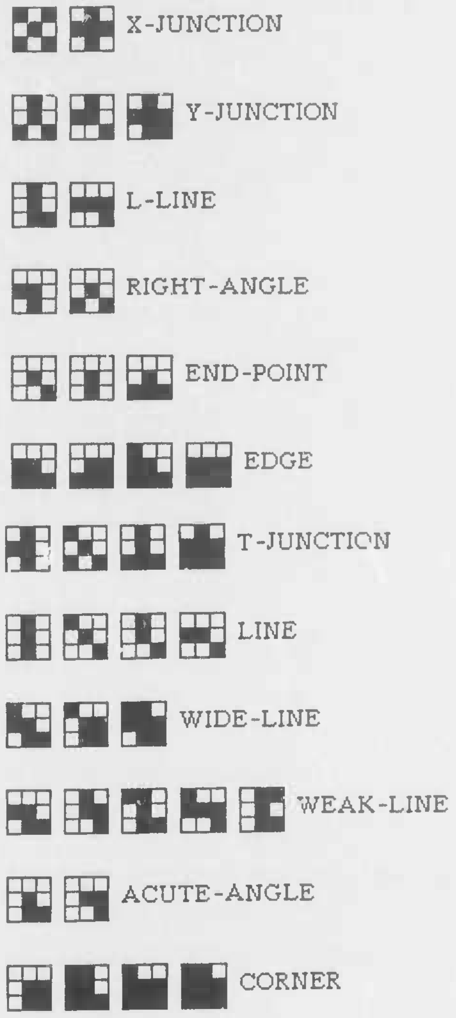
[5] McKeown, D. M. Jr., and Denlinger, J. L., "Cooperative Methods For Road Tracking In Aerial Imagery," *CMU Techical Report CMU-CS-86-175*, December 1986

[6] Neu, R., Heibler, W., Kazmierczak, H., and Sties, M., "Integration of Artificial Intelligence Concepts into the Methods for Extracting Line Objects from Monochromatic Aerial Imagery," *FIM Report No. 156*, Forschungsintitut fur Informationsverarbeitung und Mustererkennung, March 1986

[7] Netanyahu, N. S. and Rosenfeld, A., "Mask Matching for Linear Feature Detection," *Center for Automation Research Technical Report #254*, January 1987

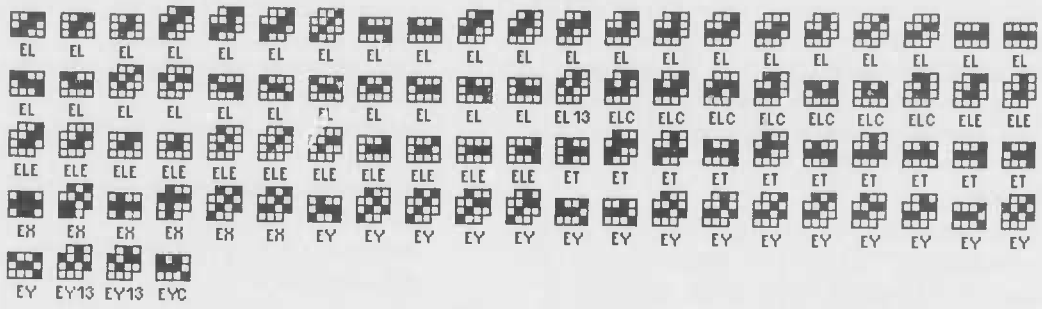
[8] Samet, H. and Tamminen, M., "An Improved Approach to Connected Component Labeling of Images," *Proc. CVPR*, June 1986, pp. 312-318

[9] Rosenfeld, A. and Kak, A. C., *Digital Picture Processing*, second edition, Academic Press, 1982

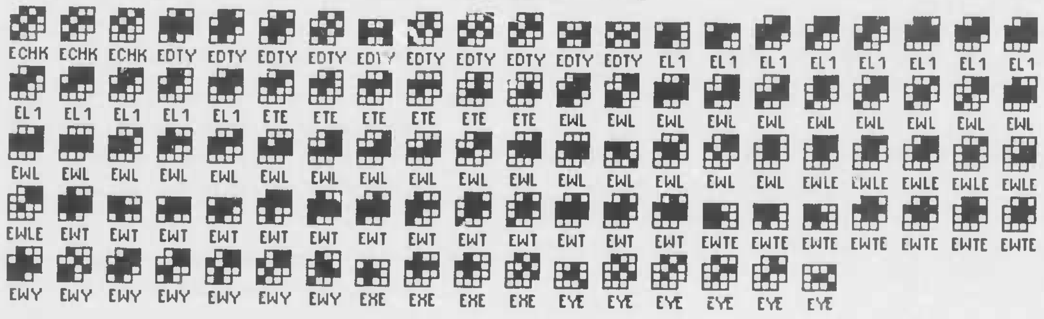


Appendix A - Binary 3×3 Masks Used

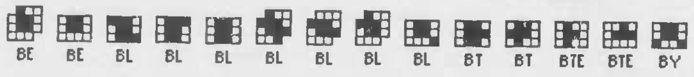
EXTEND


EL EL EL EL EL EL EL EL EL EL EL EL EL EL EL EL EL EL EL EL EL
EL EL EL EL EL EL EL EL EL EL EL EL13 ELC ELC ELC ELC ELC ELC ELC ELE ELE
ELE ELE ELE ELE ELE ELE ELE ELE ELE ELE ELE ET ET ET ET ET ET ET ET ET ET
EX EX EX EX EX EX EY EY EY EY EY EY EY EY EY EY EY EY EY EY EY
EY EY13 EY13 EYC

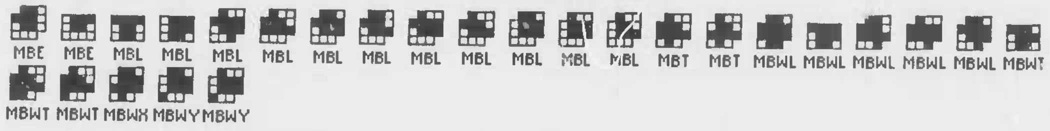
MAYBE-EXTEND


ECHK ECHK ECHK EDTY EDTY EDTY EDTY EDTY EDTY EDTY EDTY EDTY EDTY EL1 EL1 EL1 EL1 EL1 EL1 EL1 EL1
EL1 EL1 EL1 EL1 EL1 ETE ETE ETE ETE ETE ETE EWL EWL EWL EWL EWL EWL EWL EWL EWL EWL
EWL EWL EWL EWL EWL EWL EWL EWL EWL EWL EWL EWL EWL EWL EWL EWL EWLE EWLE EWLE EWLE EWLE
EWLE EWT EWT EWT EWT EWT EWT EWT EWT EWT EWT EWT EWT EWT EWTE EWTE EWTE EWTE EWTE EWTE EWTE
EWY EWY EWY EWY EWY EWY EWY EXE EXE EXE EXE EYE EYE EYE EYE EYE EYE

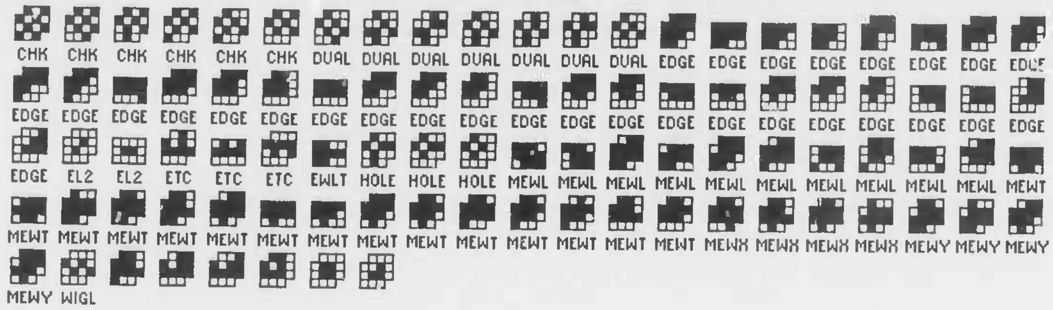
BROADEN



MAYBE-BROADEN


MBE MBE MBL MBL MBL MBL MBL MBL MBL MBL MBL MBL MBL MBT MBT MBWL MBWL MBWL MBWL MBWL MBWT
MBWT MBWT MBWX MBWY MBWY

None


CHK CHK CHK CHK CHK CHK DUAL DUAL DUAL DUAL DUAL DUAL DUAL EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE
EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE EDGE
EDGE EL2 EL2 ETC ETC ETC EWLT HOLE HOLE HOLE MEWL MEWL MEWL MEWL MEWL MEWL MEWL MEWL MEWL MEWL MEWT
MEWT MEWT MEWT MEWT MEWT MEWT MEWT MEWT MEWT MEWT MEWT MEWT MEWT MEWT MEWX MEWX MEWX MEWX MEWY MEWY MEWY
MEWY WIGL

Appendix B - Consistency Types Used

B1 The line direction is assumed to be perpendicular to the direction of the link between masks B and F as shown in Figure 4. If a mask belonging to the same connected component as B and F at either pixel P or pixel Q has at least one extending link or two maybe-extending links perpendicular to BF, then we suppress the broadening type link at both ends since it is redundant.

B2 Otherwise, we suppress the link at the mask having lower "extendability," which is a weighted sum of the number of extending type links attached to masks belonging to the same connected component at the pixel (P or Q):
extending link not perpendicular to BF – weight 2
maybe-extending link perpendicular to BF – weight 2
maybe-extending link not perpendicular to BF – weight 1

B3 If the "extendabilities" are equal, we suppress the mask that has the lower "confidence," which we chose to be the sum of mask contrast and robustness (or make an arbitrary choice if the "confidences" are equal).

E1 The line direction is assumed to be parallel ($\pm 45^\circ$) to the direction of the link between masks B and F as shown in Figure 5. If there is a parallel extending type link (as there is between masks A and D in the figure) belonging to the same connected component, we suppress both ends of the link having the lower "extendability," which is a weighted sum of the extending type links connected to A and D (and similarly to B and F). The sum used is similar to that for the "extendability" of the broadening type links, but in this case:
extending link from A or D not perpendicular to AD – weight 4
extending link from A or D perpendicular to AD – weight 2
maybe-extending link from A or D not perpendicular to AD – weight 2
maybe-extending link from A or D perpendicular to AD – weight 1

E2 If the "extendabilities" are equal, we apply rule E1 summing over all masks at pixels P and Q (and pixels R and S for the second link) that belong to the same connected component.

E3 If the new "extendabilities" are still equal, we suppress both ends of the consistency link having lower joint robustness (or make an arbitrary choice if the joint robustnesses are equal).

D1 If a mask still has an unsuppressed broadening type consistency link and there are no extending links and fewer than two maybe-extending links attached to the mask, we suppress all of the unsuppressed consistency links attached to the mask.

D2 If a pixel on the border of a connected component still contains unsuppressed masks that make the mid-line thick, we suppress all the masks at that pixel.

Appendix C - Heuristics for the suppression of masks not on the mid-line

Searching for Geometric Structure in Images of Natural Scenes

George Reynolds
J. Ross Beveridge
Department of Computer and Information Science
University of Massachusetts at Amherst
Amherst, Massachusetts 01003

Abstract

In this paper we examine the problem of grouping tokens extracted from images of natural scenes into geometrically significant components useful for image interpretation. We propose an algorithm which hypothesizes groups based on the Gestalt laws of perceptual organization, and uses an notion of simplicity in order to resolve conflicting hypotheses. Initially we are examining the problem of grouping straight line segments into larger geometric structures using the geometric relations of collinearity, parallelness, relative angle and spatial proximity. Lines may be viewed as the nodes of a graph and the geometric relations between lines as links in this graph. Significant geometric structures then arise as connected components which our results show bear a close relation to interesting scene events.

1. Introduction

Interpreting an image involves many processes of description and explanation which transform the original intensity array into a form appropriate for the goals of the system (see [10,16,28,15]). Any interpretation system must deal with two central issues: (1) How are the semantics of the scene to be defined in terms of the description and explanation processes?, and (2), How is the enormous search space, which contains the projection of the scene events of interest, to be pruned to managable size? In this paper we discuss a class of algorithms for grouping collections of primitive image events, herein called "tokens", into *geometrically* significant components useful for image interpretation.

Crucial to the process of constructing an image interpretation is the generation of intermediate level tokens which *reduces* the amount of data which needs to be processed, hence cutting the search space, while *increasing* the amount of information carried with each token and providing the semantic primitives for the interpretation processes. These tokens fit into a variety of *abstraction hierarchies*, such as spatial resolution, part - whole and, important for this paper, image geometric - scene semantic.

*This work has been supported by the following grants: DARPA N00014-82-K-0484 and DMA 800-85-C-0012

The relationship between geometric events in the image and corresponding events in the scene has attracted the attention of the image understanding community for many years. Some work has been done recently in the context of aerial photographs, see for example [5,24,13,17]. The work done in the "blocks world" domain (see [8,26,23]) includes many analyses of the relation between image geometry and scene semantics. It is interesting to note that many of the "blocks world" algorithms are good examples of the explanation processes which Witkin and Tennanbaum ([28]) assert are important to the process of image interpretation. Events such as T-junctions have only a fixed number of "explanations" within a blocks world scene, and spatially propagating these events and taking advantage of the mutual constraints between pairs of events results in a single "explanation" or 3-D structure consistent with the constraints.

One reason that this work did not generalize to natural scenes is that the events which occur in natural scenes could not be represented in terms of the primitives which the blocks world algorithms assumed as input. The work presented in this paper builds the structures which, for images of natural scenes, *require explanation* in terms of scene events.

In the context of natural scenes Hanson and Riseman ([10]) proposed a hierarchical decomposition of declarative knowledge (see Figure 1). This decomposition involved placing explicit representions of geometric events in the image at different levels of the hierarchy and linked them via a knowledge base to scene events. The bottom three levels of this hierarchy involve geometric relations in the image The algorithms presented in this paper can be viewed as manipulating the representations residing at these levels.

Of course natural scenes are rich in geometric structure, and we present here a methodolgy for constructing tokens whose features and descriptions provide the necessary cues for infering the scene structure. We will argue that this requires complex hypothesis generation and resolution strategies at the image description stage of the processing (see also [27]). The initial results reported on in this paper deal with the problem of grouping line collections of the type typically seen in road scenes and aerial photographs.

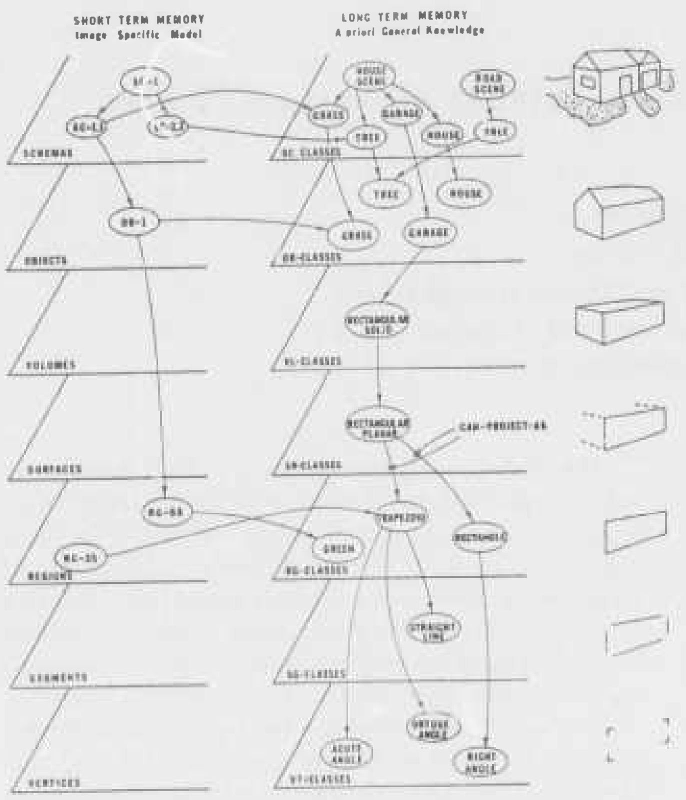
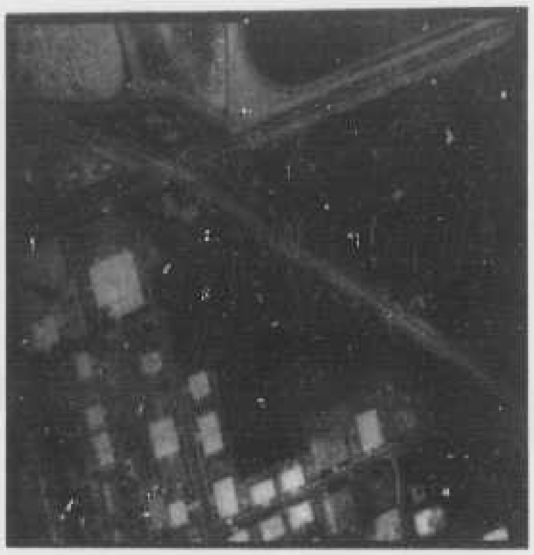


Figure 1: Hierarchical decomposition of geometric knowledge. From Hanson and Riseman 1978.



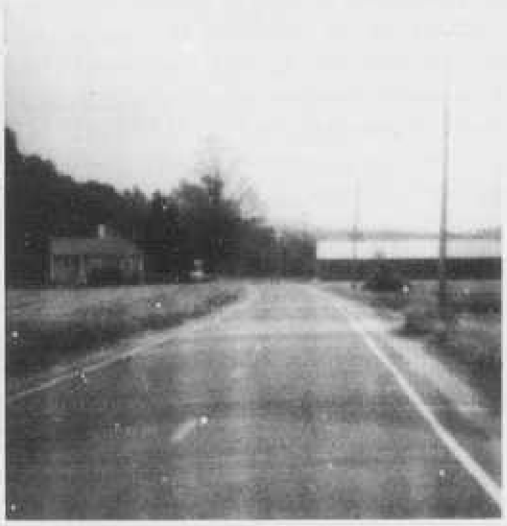


Figure 3: Natural Scene, a Rural Road.

Generally scenes of this nature will require that the geometric relations between lines and regions be combined in different ways in different parts of the image, and a general system will require a family of grouping strategies guided by a knowledge-directed or schema-based interpretation system (see [25,7]). We present some preliminary results on natural outdoor scenes and aerial photographs in which lines are grouped based rectilinearity. The results demonstrate that our system can succesfully extract geometric structures which closely correspond to important semantic features in these domains.

Let us review briefly the kind of primitives which we can assume our system will have as input. In Figure 2 we see an aerial view of and urban area and in Figure 3 we see a road scene.

Figure 4 and Figure 5 show the results of a local histogram based segmentation algorithm on the images in Figure 2 and Figure 3 respectively (see [3]). Here the input is some collection of pixels and the output is a collection of regions each of which is homogeneous with respect to some (possibly complex) feature. Figure 6 and Figure 7 show the results of a straight line extraction algorithm (see [6]) where the output lines are formed from a set of pixels of approximately uniform gradient direction. These results, although in some ways quite good, are typical of the output of complex low level algorithms. These algorithms often produce fragmented and in some cases (from the point of view of the human obeserver) incorrectly located region boundaries and straight lines.

Moreover, neither of these image abstractions alone contains the information needed to capture the semantic richness of the scene. This is true for two reasons. First, in the case of the region segmentation algorithm, although it has made explicit some semantically important regions of the image, important geometric structure is only implicit in the boundaries of the regions. In the case of the line data, some of the important geometic structure has been

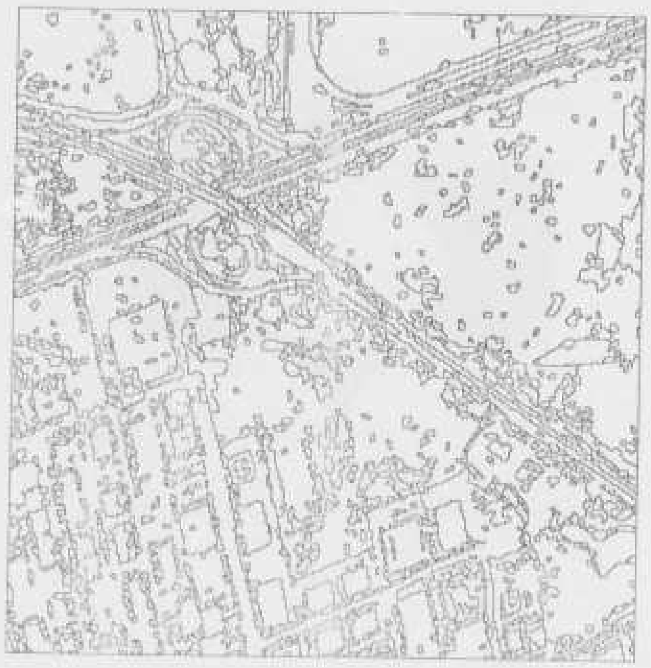


Figure 4: Region segmentation of the aerial photograph.

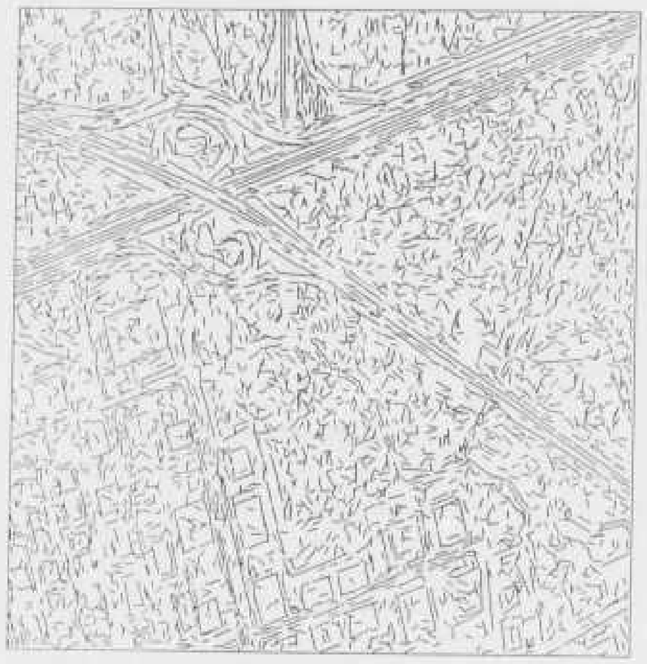


Figure 6: Straight lines found in the Aerial Photograph.

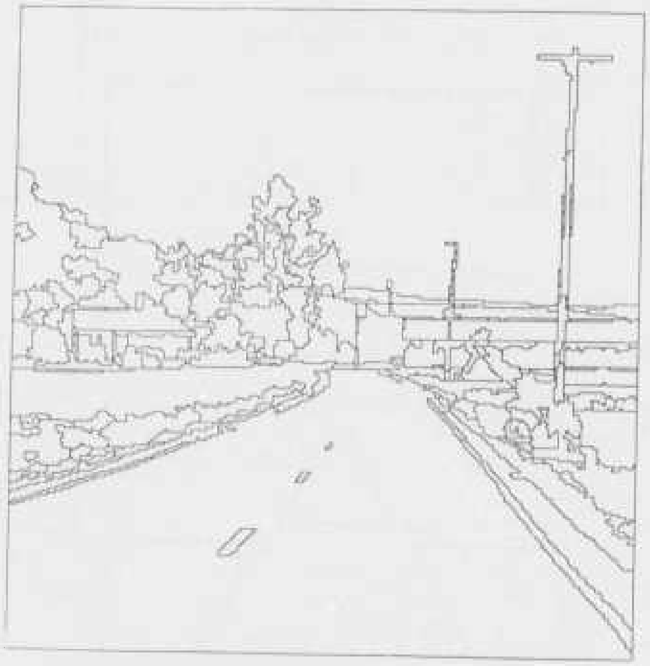


Figure 5: Region segmentation of the road scene.

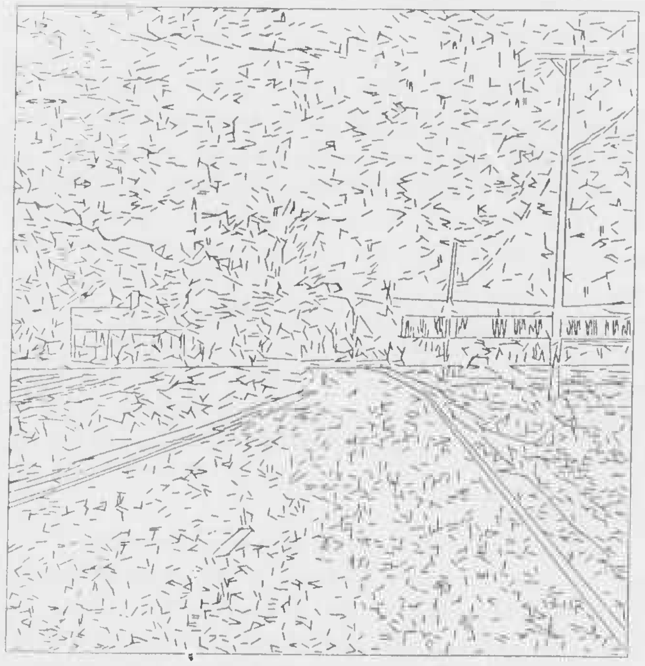


Figure 7: Straight lines found in the Road Scene.

made explicit, but more complex geometric structure such as closed regions and collections of parallel lines though immediately obvious to the human observer, are again only implicit in the lines. Secondly, regions and lines are simply not the primitives with which to represent many of the events of the scene. In the case of the road scene, the road, barn, trees and other structures are composed of groupings of these (and other) primitives. Therefore the primitves seen in Figure 4 and Figure 6 for the aerial photograph, and Figure 5 and Figure 7 for the road scene, need to be fused and manipulated in such a way that the primitives required to represent 'objects' in the scene are explicit in the resulting structres.

What is needed then are algorithms which group tokens of each type of segmentation separately and simultaneously in order to generate a more complete set of image based tokens with which to represent the image and scene events. Specifically we are interested in *geometric segmentation and grouping algorithms* where the input is some collection of tokens (regions,lines...), and the output are collections of tokens satisfying some (possibly complex) geometric relation between the tokens, for example: relations of adjacency, similar orientation, T-junction and arbitrary combinations of these involving many lines and regions.

2. Generating and Resolving Multiple Hypothesis

One difficuty in formulating any grouping algorithm or strategy is that a single token may have many possible interpretations, and in order to account for its occurence in the image, it is sometimes necessary to observe the token in more than one larger context. Thus, it is necessary to generate hypotheses which suggest possible explanations and then determine which of those hypotheses (or contexts) serves as the "best" explanation. The criteria by which such a determination is made will depend on the domain, top-down information available at the time such a determination is being made, bottom up information in the local area about the line, and various factors involving the "simplicity" (see the next section) of the structure being hypothesized.

Let us start with a simple example. Consider the image contained in Figure 8. If one is presented with a list of the lines bounding the dark region (shown in bold) and asked to organize them, and return a description of that organization, there are many possibilities. Two are shown in Figure 9 and Figure 10. Figure 9 shows one natural organization of the lines using the Gestalt Law of "good continuation" as the primary organizational principle, and suggests an addmittedly incomplete description of Figure 8 such as: "a dark figure consisting of a diamond and a rectangle, 'transparently' overlapping".

Figure 10 shows another natural organization using the Gestalt Law of "closure" as the dominant organizational

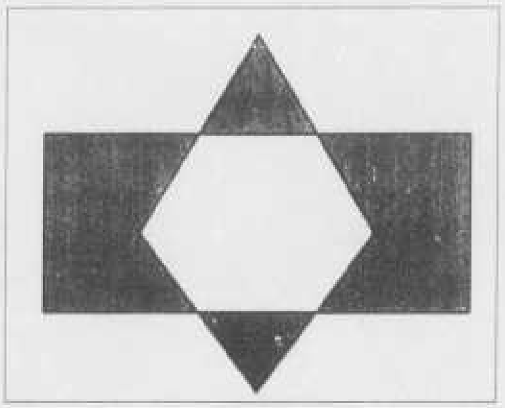


Figure 8: A grouping description problem.

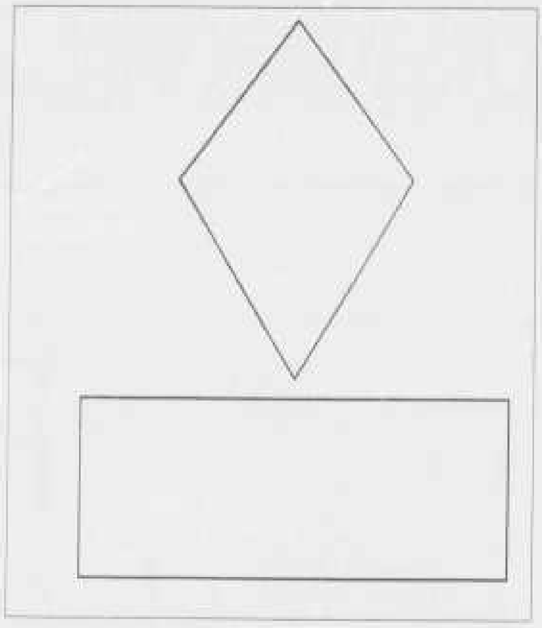


Figure 9: Description 1

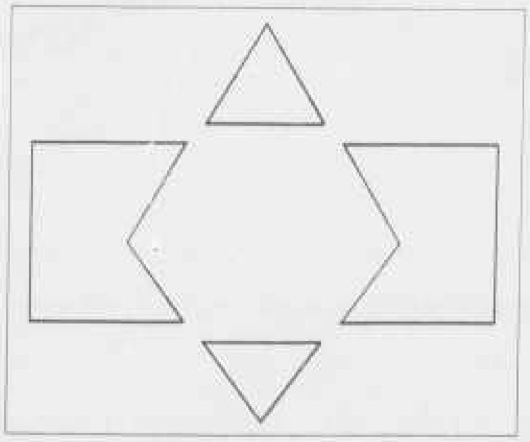


Figure 10: Description 2

principle and suggests another also incomplete description of Figure 8 such as: "a dark figure composed of two pentagonal figures and two triangles with a hexagonal hole in the middle".

From a psychological point of view the descriptions suggested in Figure 9 and Figure 10 are very different, and one might be able to determine which of these organizations (or even some other) is more "natural" to a human observer. However from the point of view of computer vision, each of these descriptions, and many others for that matter, are equally reasonable. Indeed there is no reason to prefer one over the other, *unless* there are scene (general or specific) constraints which guide the selection.

Indeed it is exactly such constraints which are at work in the human visual system. The question is, how do we translate these constraints into the computational processes of a computer vision system? In natural scenes the situations in which region and line relations allow for multiple hypotheses of this nature to be generated expand exponentially with the number of tokens. Methods for translating scene knowledge and constraints into constraints on the grouping and hypothesis generation processes are crucial to limiting the number of either top-down or bottom-up hypotheses actually formed. It is natural for the bottom-up processes to draw these constraints from measures of form (see below) applied to the groups which are generated.

In the above two examples, the notions of line, region, closure, angle, square, rectangle and other measures of form were crucial to the ultimate descriptions which resulted. Moreover these descriptions are natural and important to image interpretation for two reasons. First they lead to "simple" descriptions of the image events, within the repertoire of descriptions available and consistent with the data. Second, these descriptions provide a rich set of primitives for defining a representation of the events in natural scenes.

It has been proposed by the Gestalt school that when presented with more than one description (as in the above example) of an image event, the perceptual system tends to perfer the one involving the "simplest" description. This notion of simplicity has received much attention in the literature and has in more recent years been revised and updated by a number of authors (see [22]). One variation is from the so called empiricist point of view wherein the "most economically encoded" representation is preferred; and another is from the so called constancy point of view wherein the preference is for the description or explanation in which the "object remains constant". Rock has recently proposed another principle wherein "an executive agency seeks to explain seemingly unrelated but *co-occurring* stimulus variations on the basis of a common cause or in the case of stationary configurations, seeks solutions that explain seeming coincidences and unexplained regularities that otherwise are implicit in the nonpreferred solution" (see [Rock] page 133). For example in Figure 11 we see an example where the "executive agency" prefers the perception (description) which explains the occurence of the collinear lines over the perception (description) which yeilds (the "simpler") two symmetrical objects. Apparently the "executive agency" forces a perception explaining the collinearity over one explaining the symmetry. Perhaps the elements requiring explanation do not even include the two symmetrical objects.

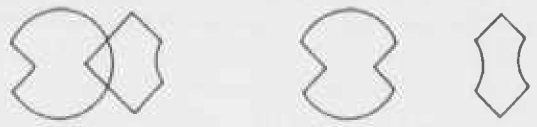


Figure 11: It is difficult to perceive (describe) the figure on the left as composed of two symmetrical figures, even though it seems to be a "simpler" description. After Rock 1983.

These proposals are interesting not only from the point of view that it helps to explain observed phenomena in human perception, but also that they provide a preliminary framework for constraining the hypothesis generation processes. For example, Julian Hochberg [1] suggested a notion of simplicity or "figural goodness" (in the context of economic encoding) which he proposed explains our perception of line drawings in three dimensions. Consider the following three features of a line drawing:

1. The number of angles enclosed within the figure.
2. the number of different angles divided by the total number of angles.
3. The number of continuous lines.

Hochberg proposed that minimizing a quantitative measure defined in terms of these features yielded the structure most likely to be perceived. Thus we perceive the object to the left in Figure 12 as a three dimensional cube "since"

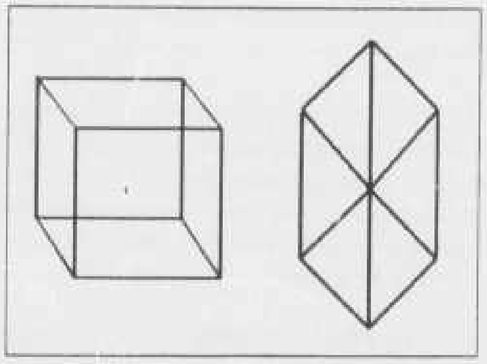


Figure 12: The Necker cube.

it minimizes Hochberg's measure. Moreover when the cube is viewed from a particular angle (the object to the right in Figure 12) we no longer perceive the three dimensional structure, "since" there is a simpler description (with respect to Hochberg's measure) which yields a two dimensional perception.

We are *not* interested here in the validity of Hochberg's measure for human perception. We are interested though, in the notion of finding the simplest description for the maximum amount of structure as a possible model for creating bottom-up organizational processes. These processes will require measures of "simplicity" guided by both top down and bottom-up information in order to significantly reduce the search space and build appropriate semantic primitives. In this paper we will develop an algorithm which uses this paradigm for line organization and apply it to line grouping in the context of natural scenes.

3. Requirements for a Geometric Grouping System

There are four essential requirements on any geometric grouping system. First is the representation of primitive tokens (in our case lines) and the geometric relations between tokens (in our case collinearity, parallelness, relative angle, and spatial proximity. Our choice of a representation will be discussed in next section.

Second, it must identify the relationship between the groups of tokens satisfying these primitive relations, and knowledge about the domain being interpreted. That is, one must select primitives which form the basis of a language for representing objects and knowledge about the scene.

Third, it must develop grouping strategies, based on the objects of interest in the scene, knowledge of the domain and the current state of the system. That is, one must build a language of *grouping strategies* for representing procedural knowledge about the objects of interest in the domain. Finally, at all stages of grouping the system must deal explicitly with the problem of search, and its relation to the objects in the domain which are to be hypothesized.

The algorithm presented in this paper is part of a larger computational framework under development at UMASS (see also [27]) for confronting the issues described above. We view the grouping and search processes as part of a 4 stage iterative grouping and extraction strategy which can be summarized as follows:

- *Primitive Structure Generation*: These processes provide the primitives (regions, lines, more complex tokens) which are the input to the grouping and hypothesis generation process described next.

- *Linked Structure Generation*: This step applies very general geometric constraints to obtain graphs within which search processes can be applied to identify specific structures of interest. For example rectilinear structures which would contain rectangles or other simple geometric structures. This is essential for generating search spaces of reasonable size.

- *Subgraph Extraction*: This step involves the extraction of specific structures "one step up" the abstraction hierarchy, and uses the linked structures to constrain the search.

- *Replacement and Iteration*: Having extracted more abstract tokens, these can now play the role of primitives in another pass of grouping and extraction.

The algorithm presented in this paper is at the *Linked Structure Generation* stage of the strategy and is applied here only to straight lines for the purpose of building rectilinear structures. In [27], a similar strategy is applied with striking results for the purpose of extracting straight lines. in general each step of the grouping and extraction strategy must apply constraints which either significantly reduce the search space and/or add important information to the descriptive power of the system.

4. Identifying Rectilinear Structure

In this section we review the system under development for the extraction of rectilinear structure and our approach to limiting the search for both general and specific geometric structure. The process starts with a set of primitives which are the the output of a straight line extraction algorithm, in this case the Burns algorithm was used (see [6]). These lines are viewed as nodes in a graph and the geometric relations of

- spatially proximate collinear
- spatially proximate parallel
- spatially proximate orthogonal
- and any subset of the above,

as relations between the nodes. These specific relations will be defined below.

Hypotheses of line groups are then generated using a connected components algorithm based on the chosen geometric links. These components, which are called *Rectilinear Line Groups*, form a new class of tokens which have emergent features and form a new level of the Geometric-Semantic abstraction hierarchy. Because different choices of the primitive geometric relations yield different rectilinear groups, a given line may participate in more than one group and so we are begining to explore various notions of "simplicity" or "preference" to resolve conflicting or competing hypotheses. Finally, specific geometric structure may be identified (rectangles, collinear and parallel structure) as subgraphs of these connected components. We refer the reader to [7] for examples of the extraction of specific geometric structure corresponding to the projection of scene events.

In defining spatial proximity one must recognize that there are essentially three types of proximity to choose from. First, there is a notion of *image-dependent proximity* wherein the metric of the image itself is used to define the distance between two tokens independent of the size of the tokens. The second is *token-dependent proximity* where the distance between two tokens is a function of the size of the tokens. For example, the length of a line might be a factor in defining a notion of parallel. Finally, there is *scene-dependent proximity* where, for example in a road scene with a camera in standard position, distance between two objects at the bottom of the image might mean something very different than for two objects at the top of the image. For the results presented in this paper we have used *token-dependent proximity* for defining orthogonal, parallel and collinear relations. Clearly this is only a start, and indeed there are many issues that we have only begun to address with regard to the relations between these three types of proximity.

The process of finding rectilinear line groups involves partitioning the lines into overlapping groups filtered with respect to orientation. These groups, which we call orthogonality ranges, contain all lines within $\pi/16$ radians (11.25 degrees) of some orientation or $\pm\pi/2$ radians of that orientation. The choice of $\pi/16$ radians results in 8 distinct orthogonality ranges. By restricting the connected components algorithm to lines from a single orthogonality range the rectilinearity of the group as a whole is guaranteed. In addition, considering each of the orthogonality ranges independently reduces the search space without limiting the resulting geometric structures and lends a degree of parallelism to the process. In Figure 13 we see the lines from Figure 6 in the orthogonality range containing the greatest number of lines. In Figure 14 we see the lines from Figure 7 in the orthogonality range about the horizontal orientation.

4.1 Spatially Proximate Orthogonal Line Segments

Figure 15 illustrates our use of token dependent proximity to determine whether two line segments **A** and **B** are spatially proximate orthogonal. Three measures contribute to determining spatially proximate orthogonality, $\Delta\theta$, t and t'. If one thinks of line segment **A** as the unit vector of a coordinate system obtained by extending **A** infinitely in both directions, then t is the value where the extension of line segment **B** intersects it. The value t' is analogus to t, except the roles of lines **A** and **B** are reversed. The value $\Delta\theta$ is simply the relative orientation between **A** and **B**.

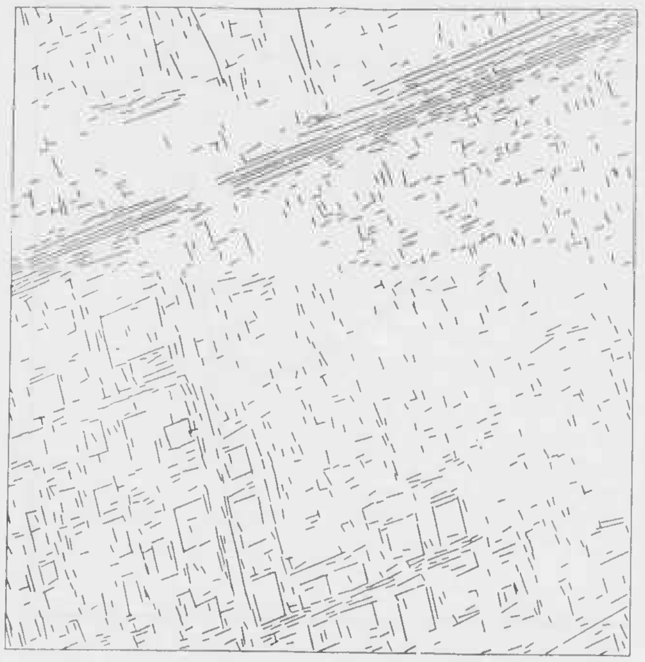


Figure 13: Lines in the Orthogonality Range for the Aerial Scene containing the greatest number of lines.

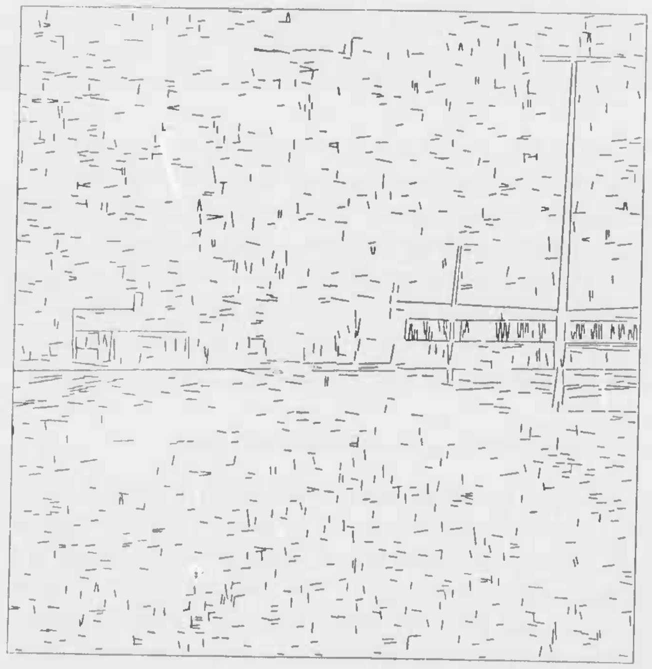


Figure 14: Lines in the Horizontal and Vertical Orthogonality Range for the Road Scene.

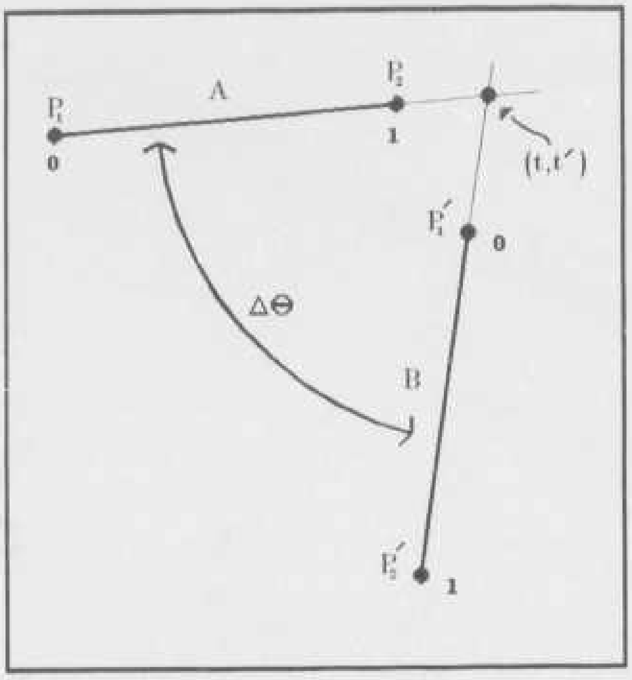


Figure 15: Illustrating Comparison of two lines for Spatially Proximate Orthogonality.

Definition: Two lines are *spatially proximate orthogonal* if the following conditions are satisfied:

- $-\delta \leq t \leq 1 + \delta$,
- $-\delta \leq t' \leq 1 + \delta$,
- $\pi/2 - \epsilon \leq \Delta\theta \leq \pi/2 + \epsilon$.

The terms ϵ and δ are thresholds which constrain the definition. For smaller ϵ the two lines must be closer to orthogonal to be related. For smaller δ the lines must be spatially closer with resepect to their endpoints. Note this definition makes no distinction between corners and T-junctions. Values of $\epsilon = 0.17$ radians (10 degrees) and $\delta = 0.5$ were used for the results in this paper and an investigation of this parameter space is currently underway.

4.2 Line Overlap and Displacement

Spatially proximate parallel and spatially proximate collinear are defined in terms of two more primitive relations which we call symmetric lateral displacement (DIS_{sym}) and symmetric overlap (OV_{sym}). The basis for these two measures are more primitive relations shown in Figure 16 where $DIS(A, B, P_1)$ measures the displacement *from* point P_1 on **A** *to* **B** and $OV(A, B)$ measures the overlap *from* **A** *to* **B**. Symmetric measures DIS_{sym} and OV_{sym} result essentially by taking averages of measures from **A** to **B** and **B** to **A** with these and the same measures with the roles of **A** and **B** reversed. A detailed explanation of how displacement and overlap are calculated can be found in [20].

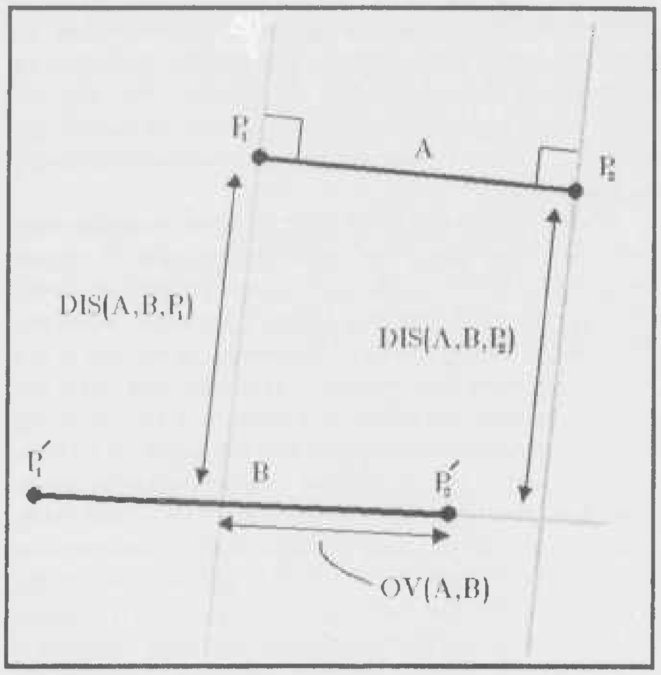


Figure 16: Illustrating Displacement and Overlap Between Two Lines.

The resulting symmetric measures satisfy the following conditions

$$0 \leq DIS_{sym}(A, B) \leq \infty$$

$$-\infty \leq OV_{sym}(A, B) \leq 1.$$

These measures are intended for use between lines already known to be roughly of the same orientation. The lateral displacement measure captures the distance in the direction orthogonal to the orientation of the line. The overlap measure captures the distance along the projection of the line. A negative overlap is used to measure the distance between two lines along their projection. For example a line completly overlaps itself ($OV_{sym}(A, A) = 1$) while there is no lateral displacement between a line and itself ($DIS_{sym}(A, A) = 0$). For two parallel sides of a square, $DIS_{sym}(A, B) = 1$ and $OV_{sym}(A, A) = 1$. For two collinear lines lying end to end, $DIS_{sym}(A, A) = 0$ and $OV_{sym}(A, A) = 0$.

4.3 Spatially Proximate Collinear and Parallel

We are now in a position to define spatially proximate collinear using the definitions of displacment and overlap.

Definition: Two lines are *spatially proximate collinear* if the following conditions are satisfied:

- $-\epsilon' \leq OV_{sym}(A, B) \leq \epsilon$
- $DIS_{sym}(A, B) \leq \delta$
- $\mid \Delta\theta \mid \leq \alpha$

For the results shown in this paper we have used the values $\epsilon' = .5$, (separated by at most 50 percent of their average length), $\epsilon = .15$, (having at most 15 percent overlap), $\delta = .15$, (displaced by at most 15 percent of their average length) and $\alpha = 0.17$, (10 degrees).

Similarly we have used displacement and overlap to define spatially proximate parallel.

Definition: Two lines are *spatially proximate parallel* if the following conditions are satisfied:

- $OV_{sym}(A, B) \geq \epsilon$
- $DIS_{sym}(A, B) \leq \delta$
- $\mid \Delta\theta \mid \leq \alpha$

For the results shown in this paper we have used the values $\epsilon = .5$, (at most 50 percent overlap), $\delta = .5$, (at most 50 percent of the average length apart), and $\alpha = 0.17$ (10 degrees).

In summary these definitions provide measures of token dependent proximity with respect to the relations of collinear, parallel and orthogonal. There is much work to be done in exploring the parameter spaces employed in these definitions. In Figures 17, 18 and 19 we see the line pairs satisfying the relations of spatially proximate orthogonal, spatially proximate collinear and spatially proximate parallel for the aerial photograph. Figures 20, 21 and 22 show the same relations for the road scene. The pairs shown in these figures are the in input to the connected components phase of the grouping algorithm which we describe in the next section.

4.4 Line Group Generation and Competing Hypothesis Resolution

We are now in a position to describe the creation of rectilinear groups. Their generation really involves two processes:

1. *Connected Components:* which forms an hypothesis of a spatially proximate rectilinear structure.
2. *Voting Processes, Competing Hypothesis Resolution:* which tries to find the simplest explanation for the largest collection of lines.

The connected components algorithm uses any subset of the three relations, spatially proximate orthogonal,spatially proximate collinear and spatially proximate parallel. These different subsets make up a family of possible relations upon which the connected components can be based. For example we might choose to select conditions only from the rules for collinearity and parallelness and ignore the orthogonality conditions. The output of the connected components algorithm would be structures all of whose elements are parallel or collinear with the property that any line in the group is spatially proximate (in the sense described above) to at least one other line in the group.

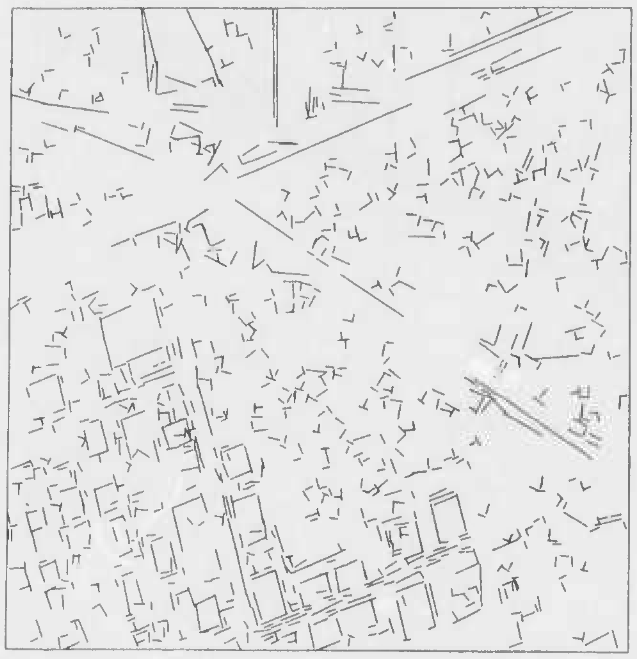


Figure 17: Lines Belonging to Spatially Proximate Orthogonal Pairs for the Aerial Photograph.

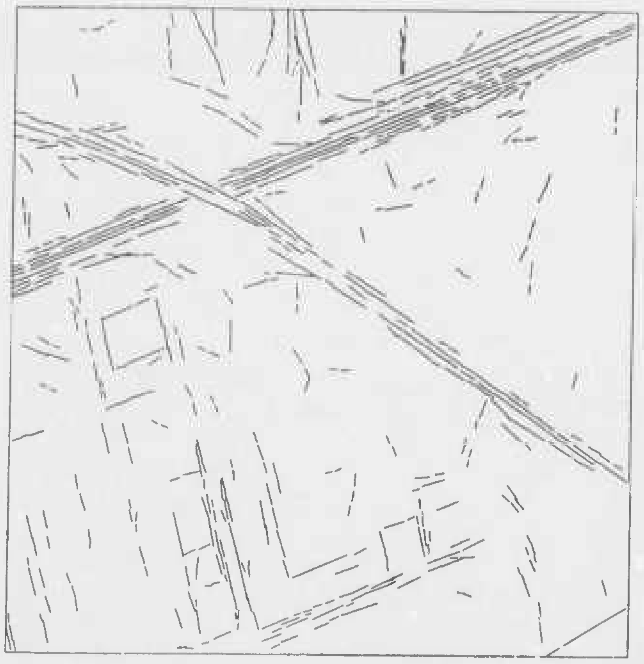


Figure 18: Lines Belonging to Spatially Proximate Collinear Pairs for the Aerial Photograph.

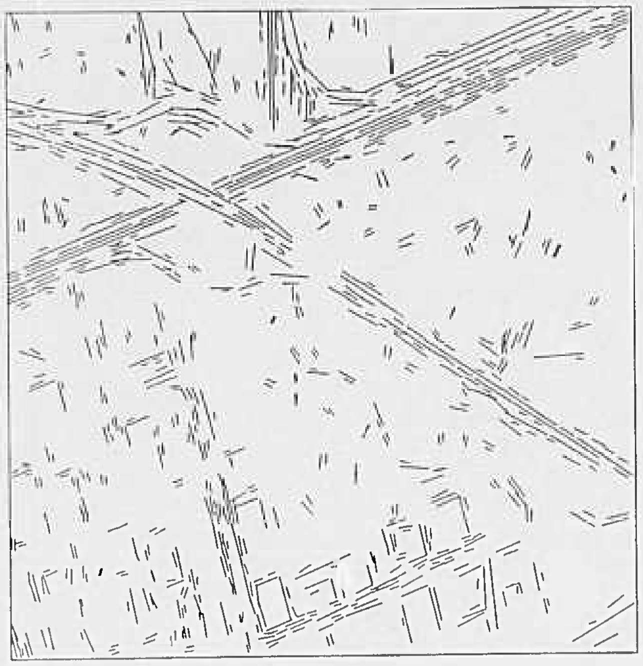


Figure 19: Lines Belonging to Spatially Proximate Parallel Pairs for the Aerial Photograph.

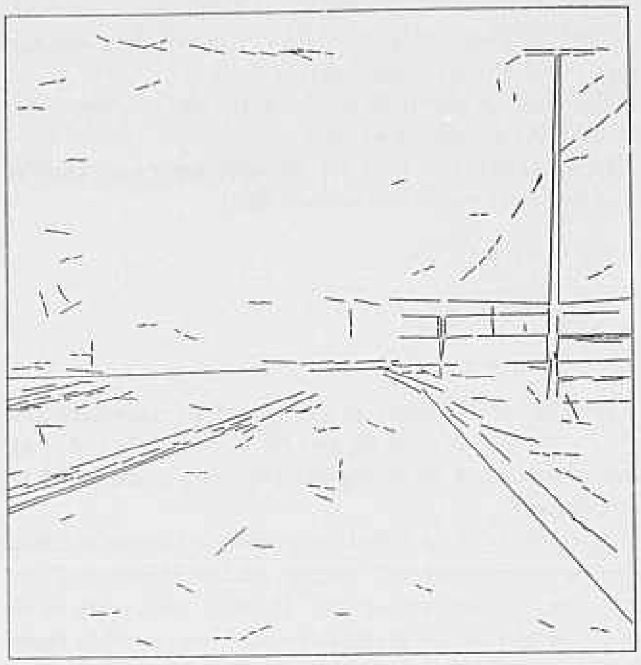


Figure 21: Lines Belonging to Spatially Proximate Co-linear Pairs for the Road Scene.

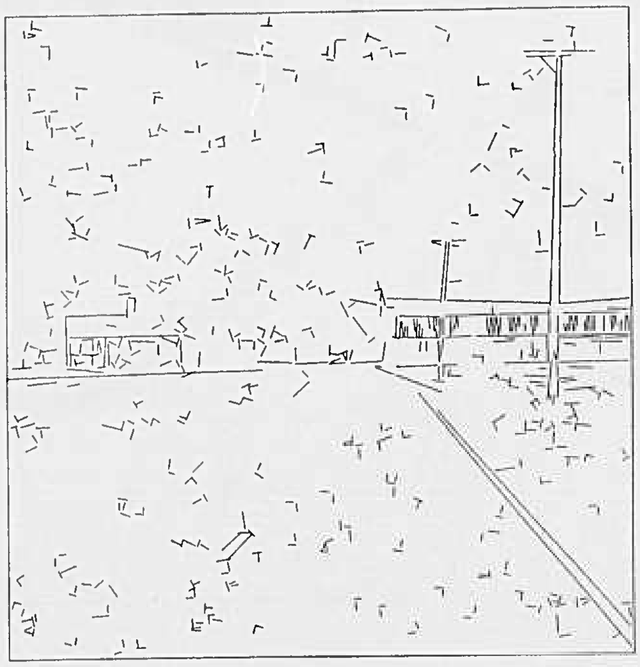


Figure 20: Lines Belonging to Spatially Proximate Orthogonal Pairs for the Road Scene.

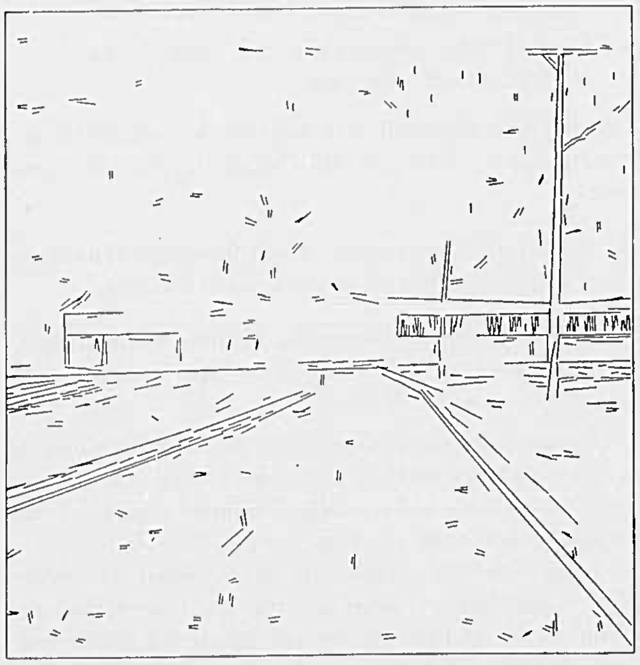


Figure 22: Lines Belonging to Spatially Proximate Parallel Pairs for the Road Scene.

Given that the the connected components algorithm is run separately on each of the othogonality ranges and that the orthogonality ranges overlap, each line can belong to two connected components. In addition, groups based on different combinations of links are formed separately and a line can belong to groups of each type independently. Currently we have explored resolution within groups formed using a single combination of links. One means for accomplishing this resolution is to invoke a process which asks of each line which component it prefers according to various selection criteria. For example: Let each line vote for the group which has the most lines, or the group whose total length is longest, or the group which is "simplest" based on some more complex rule. Some preliminary results are shown in the next section.

5. Results

The Rectilinear Line Grouping System, as the system described above is called, has been run on roughly ten images drawn from different domains including road scenes, aerial photographs and house scenes. In each case connected components, or what we will simply call line groups, were formed using all combinations of the three basic relations, spatially proximate orthogonal, spatially proximate collinear and spatially proximate parallel. These line groups clearly represent a next step up the geometric-semantic abstraction hierarchy. The groups we have produced differ greatly in size and form, capturing a wealth of structure in the images. We are just begining the process of learning how best to utilize and characterize these groups. Presented here is a sampling of the groups generated for the images presented in Figures 2 and 3.

5.1 A Sample of the Connected Components Grouping Results

In Figure 23 we see the lines which belong to the group containing the largest number of lines formed using only the relation spatially proximate orthogonal. The source lines (Figure 6) are for the aerial photograph. This group contains 82 lines, the next largest group of the same type for this image contained only 19. An examination of Figure 17 shows that many spatially proximate orthogonal line pairs are identified around the buildings in the lower lefthand portion of the image. The connected components grouping finds a number of groups containing roughly 5 to 15 lines in this area.

The line groups shown in Figure 24 are the result of grouping lines from the same aerial photograph based on the relations spatially proximate collinear and spatially proximate parallel. The figure was generated by selecting the 4 largest groups and displaying 'best' 3. How the best 3 were chosen will be discussed below as an example of resolving multiple hypothesis.

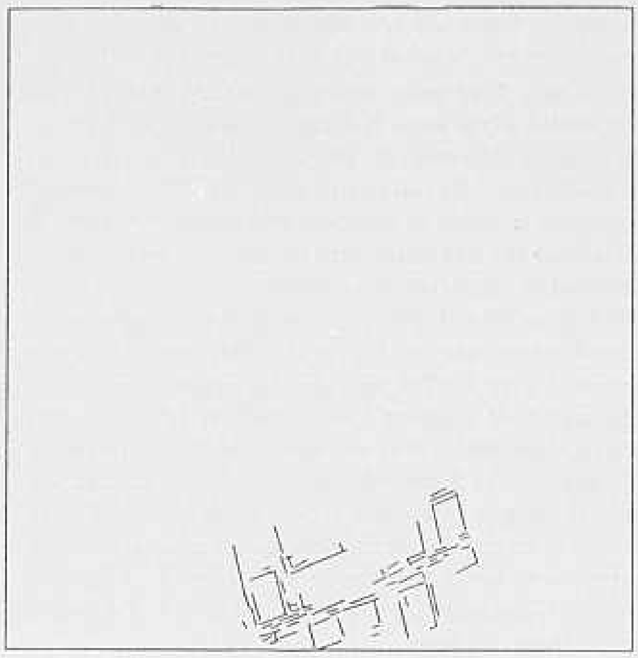


Figure 23: Largest Group Based on Spatially Proximate Orthogonality.

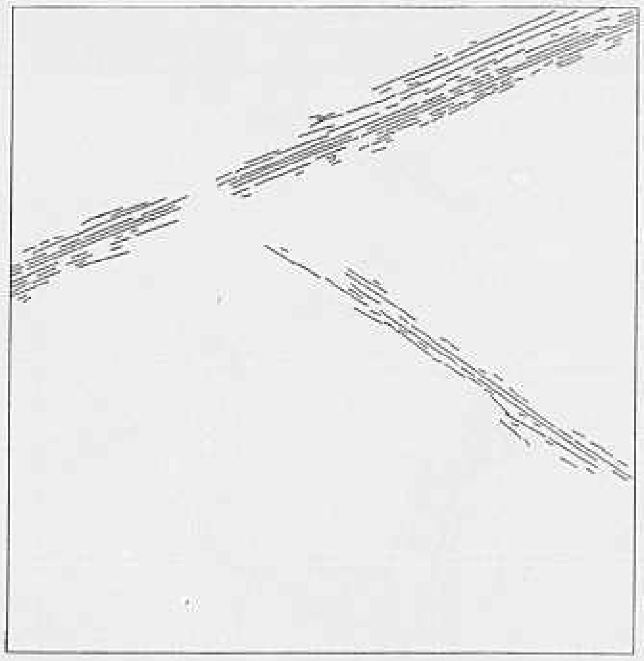


Figure 24: Largest Groups Based on the Relations Spatially Proximate Collinear and Spatially Proximate Parallel.

The line groups shown in Figure 25 were produced using all three relations, spatially proximate orthogonal, spatially proximate collinear and spatially proximate parallel. Four groups are shown, selected out of the largest 5. What appears to be a large group resulting from buildings in the lower portion of the scene is actually two groups. The two groups contain 158 and 135 lines respectively and share 54 lines in common. The two groups arose out of the connected components grouping in adjacent orthogonality ranges. It is clear that the significant groups based on rectilinearity correspond to significant scene events.

In Figures 26 and 27 we see rectilinear line groups for the road scene image of Figure 3. The source lines are shown in Figure 7. The two groups shown in Figure 26 are the result of grouping based solely on spatially proximate orthogonality. They are the 2 'best' groups out of the largest 4. The 2 discarded each contain about half the lines in the largest group, and are the result of grouping the lines for the large group in the orthogonality ranges adjoining the one in which the large group is found. Figure 27 shows the single largest group produced by grouping based upon all three relations.

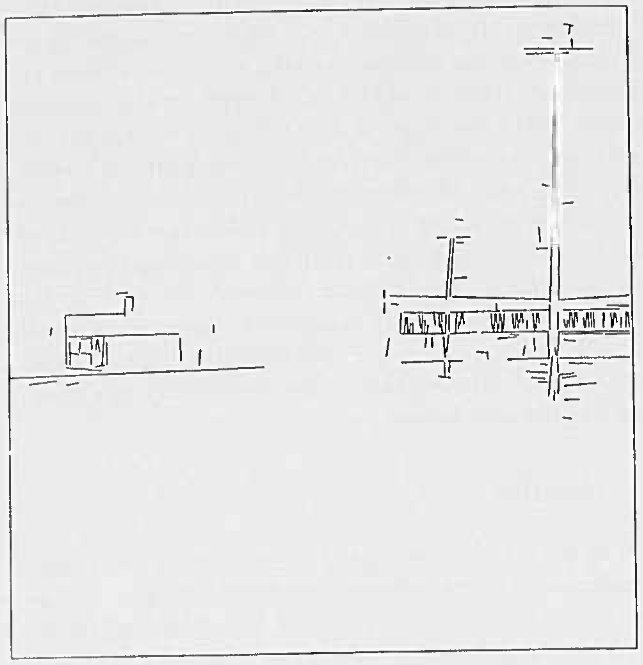


Figure 26: Largest Groups Based on Spatially Proximate Orthogonality.

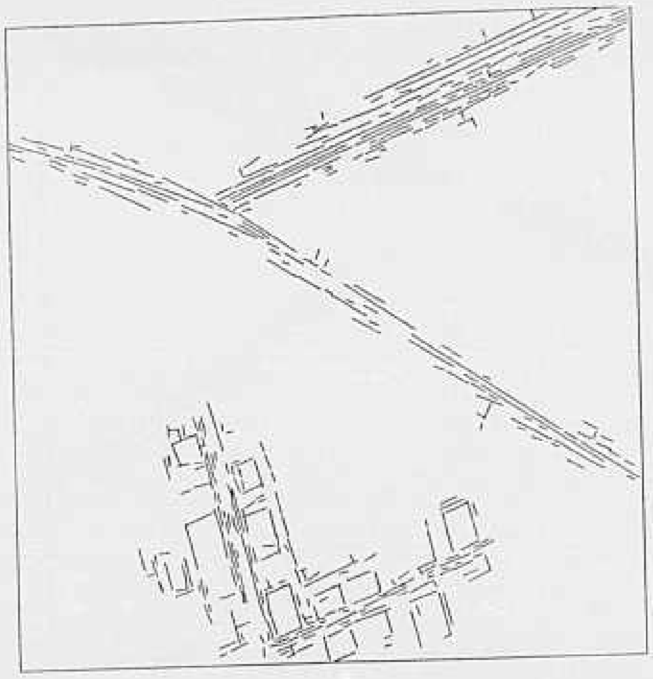


Figure 25: Largest Groups Based on all Three Relations: Spatially Proximate Orthogonal, Spatially Proximate Collinear and Spatially Proximate Parallel.

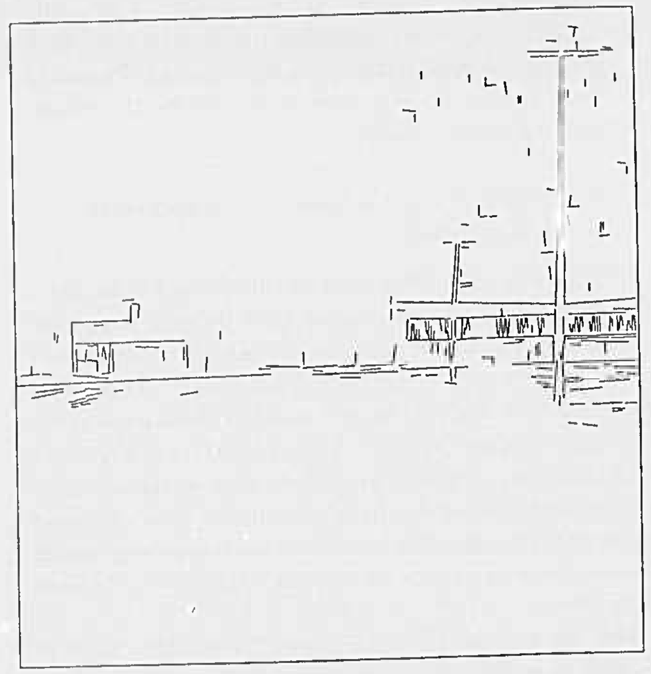


Figure 27: Largest Single Group Based on all Three Relations: Spatially Proximate Orthogonal, Spatially Proximate Collinear and Spatially Proximate Parallel.

5.2 Illustrating the Resolution of Multiple Hypothesis

In selecting the groups shown in Figure 24 we said the 'best' three of the four groups were selected. Figure 28 shows the group the was rejected and Figure 29 the group that caused it to be rejected. As mentioned earlier, a simple means of resolving conflicting hypotheses is for each line to vote for the group it 'prefers'. One basis for preference is size. Reliance on size amounts to an elementary definition of 'simplicity' which in this case amounts to the selecting the group accounting for the maximum ammount of structure. Size may be measured in many ways, and two simple ways are in terms of the number of lines or the cumulative length of the lines. The group shown in Figure 28 contains 88 with a cumulative length of 1116. The group shown in Figure 29 contains 117 lines with a cumulative length of 1336. These 2 groups share 83 lines in common. Using either of these measures, the group in Figure 29 would clearly receive the majority of the votes.

In resolving the conflict between the groups in Figures 28 and 29 it made little difference what measure of size was used. This may not always be the case, however as the groups illustrated in Figures 30 and 31 the group in Figure 31 was chosen over the one in Figure 30 for inclusion in Figure 25 by hand Which of these two groups would be considered 'best' depends upon the measure of size. The group in Figure 30 contains 107 lines with a cumulative length of 902. The group in Figure 31 contains 102 lines with a cumulative length of 983. The two groups have 70 lines in common. Hence which group is chosen depends upon the measure used. Based on cumulative length, the group in Figure 31 wins. Based on the number of lines, the group in Figure 30 wins. This example gives some flavor for the types of difficulties surrounding the issue of multiple hypothesis resolution.

6. Conclusion and Future Directions

In this paper we have examined the problem of grouping tokens extracted from images of natural scenes into geometrically significant components useful for image interpretation. An implementation of a system for grouping straight lines into larger rectilinear configruations is described. In designing and implementing this system, two central issues in geometric grouping had to be addressed. First, the importance of building structures which can serve as primitives for defining scene events and second, the importance of pruning the enormous search spaces which contain the projections of the scene events of interest. The Gestalt Laws of perceptual organization and in particular many of the rules of simplicity and economic encoding provide a framework

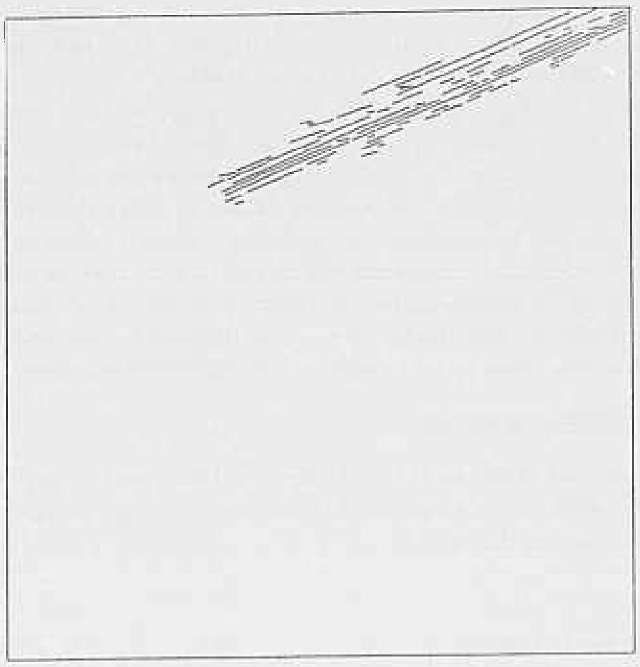


Figure 28: Line Group Based on Spatially Proximate Collinearity and Parallelness, rejected For Lack of Support.

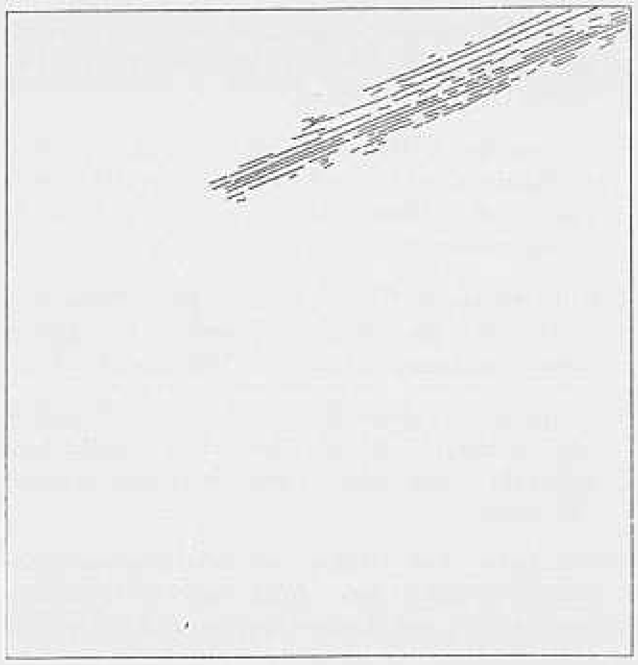


Figure 29: Line Group Based on Spatially Proximate Collinearity and Parallelness. Selected Over Group Shown the Previous Figure. Contains 117 Lines.

for developing descriptions and algorithms which constrain the number of hypothesis generated.

In this paper we have dealt only with line organization, and our future work includes extending the algorithms we describe to include line and region relations. For example in the algorithm described in this paper we have not used the direction of the intensity gradient across the line, color information in a neighborhood about the line or information about regions of a segmentation to which the line is adjacent, to constrain the grouping processes. Each of these additional constraints will allow the generated structures to be more complete descriptions of some local area in the image, allow further pruning of the search space, and provide a richer set of primitives for higher level processes.

Acknowledgements

We would like to thank Al Hanson, Bruce Draper and Michael Boldt for many valuble discussions during the development of this work.

References

[1] R. Arnheim, *Art and Visual Perception, A Psychology of the Creative Eye*, University of California Press, Berkeley, 1974.

[2] R. Belknap, E. Riseman, and A. Hanson, "The Information Fusion Problem and Rule-Based Hypotheses Applied To Complex Aggregations of Image Events," *Proc. DARPA IU Workshop*, Miami Beach, FL, December 1985.

[3] R. Beveridge, A. Hanson, and E. Riseman, "Segmenting Images using Localized Histograms," COINS Technical Report, University of Massachusetts at Amherst, in preparation, 1987.

[4] I. Biederman, A. Glass, and E.W. Stacy, "Searching for Objects in Real-World Scenes", *Journal of Experimental Psychology* Vol. 97, No. 1, 1973, pp. 22-27.

[5] R. Brooks, "Symbolic Reasoning Among 3-D Models and 2-D Images," *STAN- CS-81-861*, and **AIM-343**, June 1981, Department of Computer Science, Stanford University.

[6] J.B. Burns, A.R. Hanson, and E.M. Riseman, "Extracting Straight Lines," *IEEE Transactions on Pattern Analysis and Machine Intelligence 8*, No. 4, July 1986, 425-455.

[7] B. Draper, R.Collins, J Brolio, J. Griffith, A. R. Hanson, E. Riseman, "Tools and Experiments in the Knowledge Directed Interpretation of Road Scenes", this proceedings.

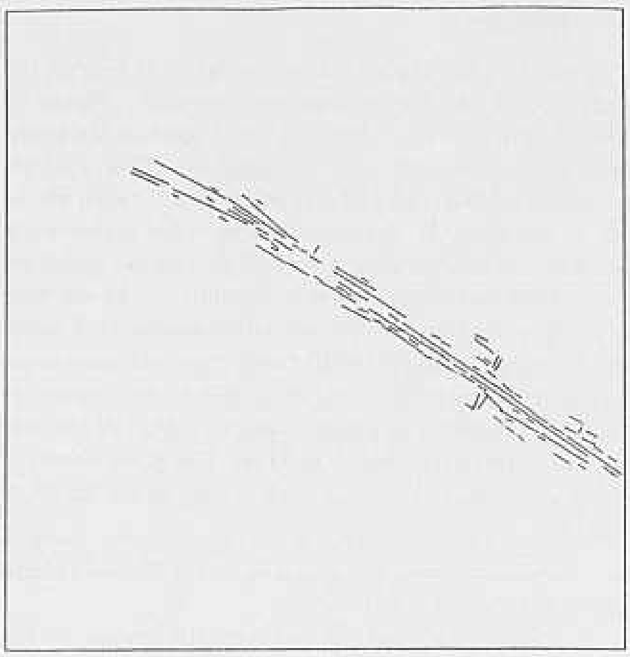


Figure 30: Line Group Based on Spatially Proximate Othogonality, Collinearity and Parallelness. Group Contains 107 lines with a cumulative length of 902.

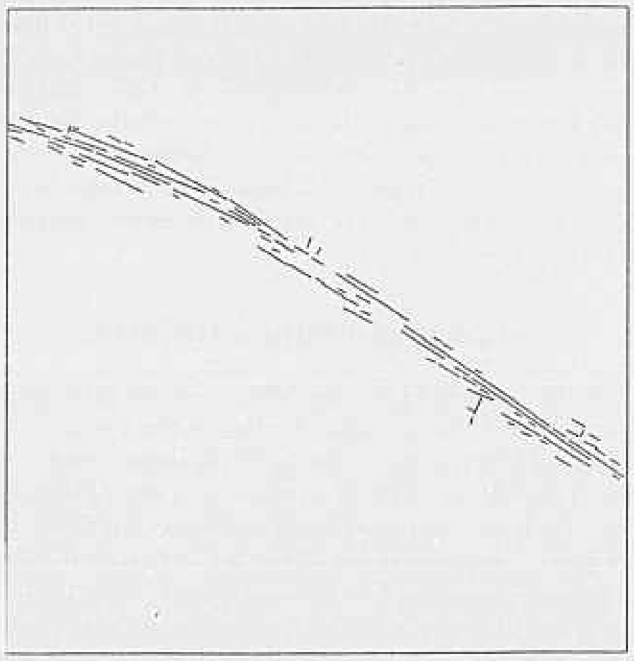


Figure 31: Line Group Based on Spatially Proximate Othogonality, Collinearity and Parallelness. Group Contains 102 lines with a cumulative length of 983.

[8] A. Guzman, "Decomposition of a Visual Scene into three-dimensional bodies", in A Grasselli, ed., *Automatic Interpretation of and Classification of Images*, Academic Press, 1969.

[9] A.R. Hanson and E.M. Riseman (Eds.), *Computer Vision Systems*, New York, Academic Press, 1978.

[10] A. Hanson, and E. Riseman, "VISIONS: A Computer System for Interpreting Scenes, in *Computer Vision Systems* (A. Hanson and E. Riseman, eds.) pp. 303-333, Academic Press, 1978.

[11] A. Hanson and E. Riseman, "Segmentation of Natural Scenes," in *Computer Vision Systems,* (A. Hanson and E. Riseman, Eds.), Academic Press 1978, pp. 129-163.

[12] A.R. Hanson and E.M. Riseman, "A Methodology for the Development of General Knowledge-Based Vision Systems," to appear in *Vision, Brain, and Cooperative Computation,* (M. Arbib and A. Hanson, Eds.) 1986, MIT Press, Cambridge, MA.

[13] M. Herman and T. Kanade, "The 3D MOSAIC Scene Understanding System: Incremental Reconstruction of 3D Scenes from Complex Images", *Proceedings of the DARPA IU Workshop*, October 1984, pp. 137-148.

[14] G. Kaniza, W. Gerbino, "Convexity and Symmetry in Figure Ground Organization", in M. Henle, ed., *Vision and Artifact*, Springer, New York, 1976.

[15] D. G. Lowe, *Perceptual Organization and Visual Recognition*, Kluwer Academic Publishers, 1985.

[16] D. Marr, *VISION*, W.H. Freeman and Company, San Francisco, 1982.

[17] D. McKeown and Pane, "Alignment and Connection of Fragmented Linear Features in Aerial Imagery", CMU Tech Report, 1985.

[18] M. Nagao and T. Matsuyama, "A Structural Analysis of Complex Aerial Photographs," Plenum Press, New York, 1980.

[19] R. Nevatia and K.R. Babu, "Linear Feature Extraction and Description," *Computer Graphics and Image Processing*, Vol. 13, 1980, pp. 257-269.

[20] G. Reynolds, J. Ross Beveridge, "Geometric Line Organization Using Spatial Relations and a Connected Components Algorithm", COINS Technical Report, University of Massachusetts at Amherst, in preparation, 1987.

[21] G. Reynolds, N.Irwin, A. Hanson and E. Riseman, "Hierarchical Knowledge- Directed Object Extraction Using a Combined Region and Line Representation," *Proc. of the Workshop on Computer Vision: Representation and Control,* Annapolis, Maryland, April 30 - May 2, 1984, pp. 238-247.

[22] I. Rock, *The Logic Of Perception*, MIT Press, Cambridge, 1983.

[23] K. Sugihara, *Machine Interpretation of Line Drawings* MIT Press, Cambridge, Mass 1986.

[24] M. Tavakoli, A. Rosenfeld, "Building and Road Extraction from Aerial Photographs", *IEEE Transactions on Systems, Man and Cybernetics* vol 12, 1982.

[25] T.E. Weymouth, "Using Object Descriptions in a Schema Network For Machine Vision," Ph.D. Dissertation, Computer and Information Science Department, University of Massachusetts at Amherst. Also COINS Technical Report 86-24, University of Massachusetts at Amherst, 1986.

[26] D. Waltz, " Generating Semantic Descriptions from Drawings of Scenes with Shadows", in P. Winston ed., *The Psychology of Computer Vision*, McGraw-Hill, New York, 1975.

[27] R. Weiss, M. Boldt, "Geometric Grouping of Straight Lines", Proc. IEEE Conf. Computer Vision and Pattern Recognition, Miami Beach, June 1986.

[28] A. P. Witkin and J.M. Tenenbaum, "What Is Perceptual Organiztion For?", *IJCAI*, 1983, pp. 1023-1026.

Detecting Runways in Aerial Images[1]

A. Huertas, W. Cole and R. Nevatia

Institute for Robotics and Intelligent Systems
School of Engineering
Powell Hall of Engineering
University of Southern California
Los Angeles, CA 90089-0273

ABSTRACT

We are pursuing the detection of runways in aerial images as part of a project to automatically map complex cultural areas such as a major commercial airport complex. This task is much more difficult that appears at first. We use a hypothesize and test paradigm. Hypotheses are formed by looking for instances of long rectangular shapes, possibly interrupted by other long rectangles. We use runway markings, mandated by standards for runway construction, to verify our hypotheses.

Keywords: Aerial image analysis, shape analysis, mapping, runway detection, Image Understanding Systems.

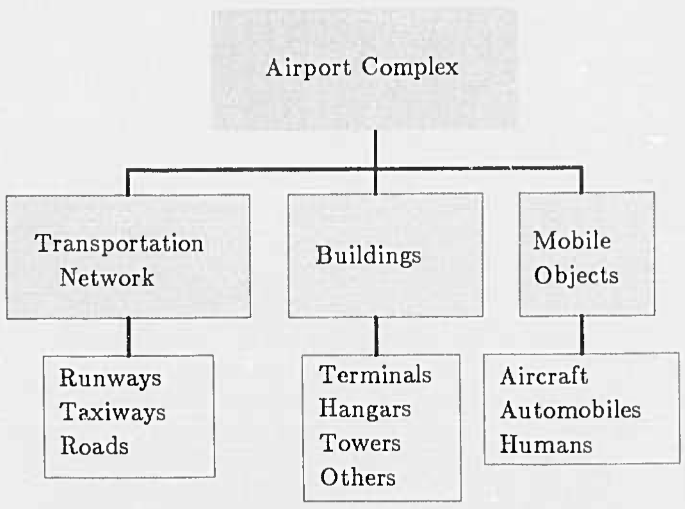


Figure 1: Objects in Scenes of Airport Complexes

1 INTRODUCTION

Our aim is to develop general techniques for automated mapping and photointerpretation tasks. We have chosen major commercial airports as a test domain that has a variety of interesting characteristics.

Airports contain a variety of objects, depicted in figure 1, such the transportation network (runways, taxiways, and roads), a variety of building structures (hangars, terminals, storage warehouses), and a variety of mobile objects (automobiles, aircraft, humans). Further, the airport complexes are under continual changes, usually due to expansion. The images themselves are rather complex due to the large number of objects present in them. The mapping of this domain, thus, offers a variety of challenging problems.

Our goal is to map all of the interesting objects in the scene and also to devise integrated descriptions that include the functional relationships of the objects in the scene. In this paper we concentrate on the mapping of runways (we are pursuing mapping of buildings in separate work [4,2]). The runways and taxiways may appear to be modeled easily – namely as long, thin, rectangular strips of uniform brigthness. However, the real images are much more complex, as shown in figure 2, a portion (LOGAN1:800 $\times$ 2200 resolution) of Logan International Airport in Boston, and in figure 3, a portion (JFK2: 2500$\times$2000 resolution) of John F. Kennedy International Airport in New York. These help illustrate the following:

- *Object complexity:* Runways have a variety of markings, some are standard as in LOGAN1, some are not (left runway in JFK2), that are applied to the paved areas of runways and taxiways to identify clearly the functions of these areas, and to delimit the physical areas for safe operation and aid pilots. In many cases there are visible signs of heavy use, such as tire tread marks, oil spots, and exhaust fume smears. Also, runways have shoulders of various widths.
- *Object composition:* Runways may not be of uniform material (JFK2). The landing surface and the shoul-

[1]This research was supported by the Defense Advanced Research Projects Agency, monitored by the Air Force Wright Aeronautical Laboratories under contract F33615-84-K-1404

ders may be of the same or different material for different runways in the same airport. Runways may be extended using different materials. In certain geographical locations, the runway surfaces develop defects that need to be repaired; the repair work, usually in the form of patches is not necessarily homogeneous with the original surface material, and can have random shapes.

- *Object functionality:* Runway surfaces may be occluded by trucks and aircraft. Runways have access taxiways and service roads in a variety of positions with respect to the runway. Runways can intersect with other runways. Also, old runways or portions of them may be now used for other purposes.

the surfaces, according to FAA specifications. From airport engineering we also know the range of angles between runways, typical distances between parallel runways, range os widths and so on.

- *Photometric Knowledge*: Intensity data may be of some help in verifying runway hypotheses when runway markings are non existent or not available due to lack of contrast or lack of resolution. Our current implementation does not make use of this knowledge but only uses the image resolution information.

In work reported here, our verification step consists only of finding the various markings we expect. We have not yet combined the different criteria to give an overall confidence value. This process should, ideally, take place in the context of the larger system that is also reasoning about other objects in the scene, such as the remainder of the transportation network, buildings and the mobile objects. Location of these objects will mutually affect the confidence levels of the descriptions of other objects. Thus, the system described here should be viewed as a module for the larger system to operate on.

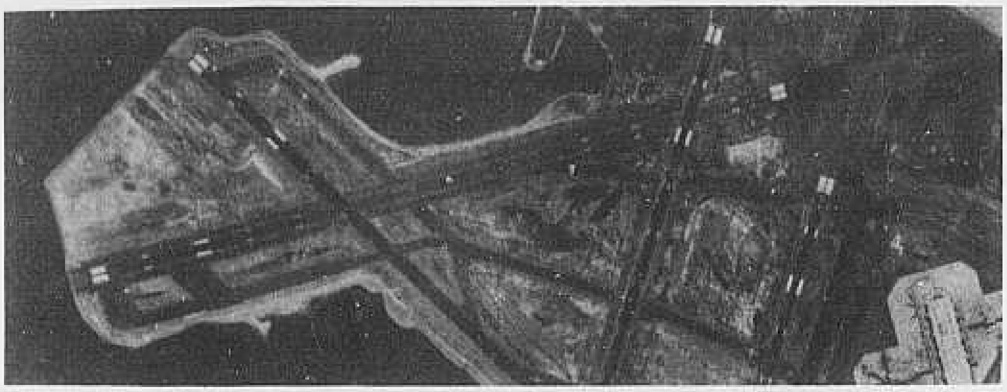


Figure 2: Logan International Airport image (LOGAN1)

One of the major causes of difficulties in detecting runways and other objects in real aerial scenes is that the low level segmentation rarely give complete and accurate results. In our work we have chosen to work primarily with the line *segments* computed from the intensity edges in the image. These lines may be fragmented, due in part to inadequacies in the line detection process, and in part due to actual structures in the image. In general, we assume that the images are of fairly good quality and of adequate resolution.

Our method uses the hypothesis formation and verification paradigm to detect runways. Our approach uses a generic model of the objects of interest derived from the following sources of knowledge:

- *Geometry* and *Shape*: We know that we are looking for instances of objects whose outlines represent a rectangular shape having a large aspect ratio of length to width. We know that runways have *ends* as opposed to nearby straight stretches of highways and roads.
- *Specific Knowledge* of airport design: We know the features that make a visible long strip in the image an airport runway: The standard markings applied to

2 AN OVERVIEW AND ASSUMPTIONS

As previously mentioned, we have chosen to work primarily with line *segments* extracted from the image. Figure 4 shows the segments computed from the LOGAN1 image in figure 2. Geometric knowledge of the desired structures indicate that they should be characterized by parallel lines of opposite contrast. We call such pairs of lines "anti-parallel", and abbreviate them as *apars*. Apars form the basic unit of our further analysis.

In a complex scene like a major airport not all apars correspond to a runway or even to segments of the transportation network. Conversely, not all of the runway is likely to be included in one apar or even in a set of apars.

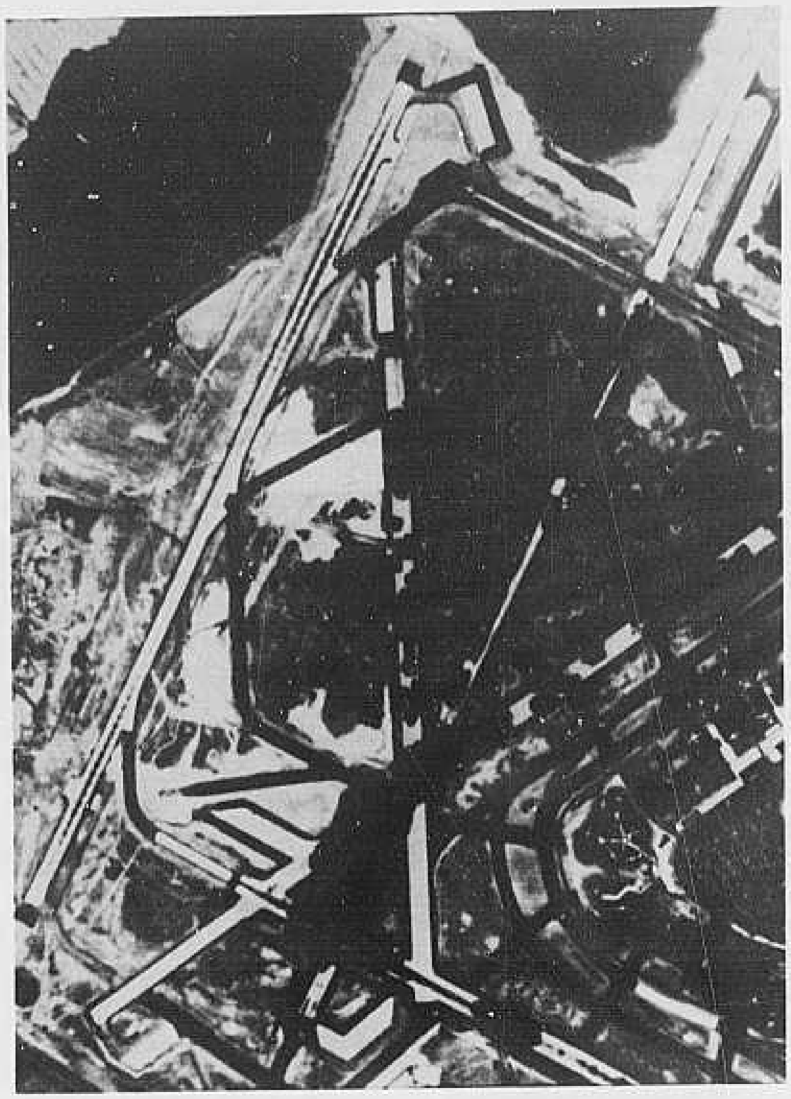


Figure 3: John F. Kennedy International Airport image (JFK2)

We need to use the constraints provided by the scene knowledge to properly interpret the apars and to make sense out of the fragmented information provided by the lower level processing. To do this we follow a hypothesize and test paradigm.

One important aspect of our hypotheses formation process is that it is as non-committal as possible. Thus, if a line *segment* contributes to many *apars*, as is the case along runway features where there may be a large number of linear features parallel to the runway, such as runway shoulders, taxiways, and service roads, possibly having markings of their own, we allow the *apar* computation process to generate all the possible combinations. This leads to a large search space that must be reduced in order to facilitate the detection of "targets" in the presence of a large number of "distractors".

Reduction of the search space is accomplished in our method by computing an estimate of the directions of runways, as well as an estimate of the widths of the runways. We assume that at least a portion of the runways is visible in the image from which we can compute these estimates. Once these estimates are available we can extract from the set of apars those having one of the selected directions and having a range of widths, and form a set of apars presumably representing fragments of runways. Note that a reduction in the search space *does not* imply a reduction of the information space. At all times the entire set of apars and segments is available.

Hypotheses formation is still a general methodology; knowledge specific to the objects of interest is required for complete disambiguation. Verification of runways is accomplished primarily by detection and identification of runway markings. Figure 5 shows the set of markings that we look for to verify commercial runways. They also help classify the runways into three basic categories: *precision instrument* runways, *non-precision instrument* runways, and *visual* runways. These in turn, tells us something about the

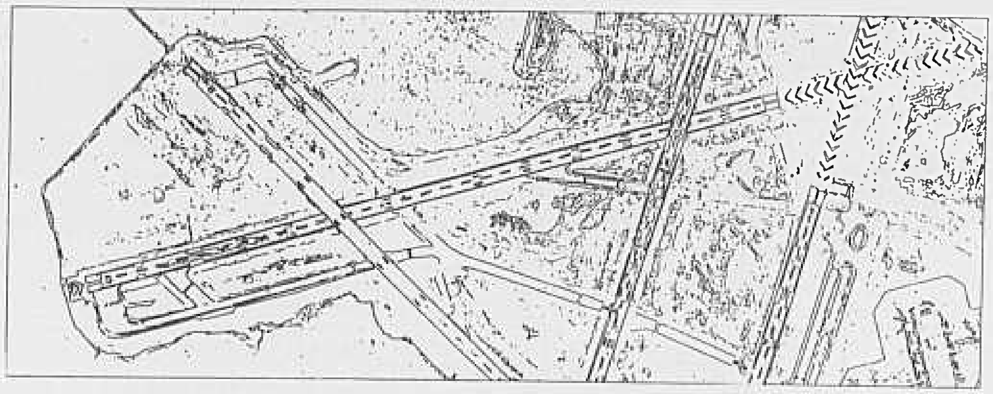


Figure 4: Line Segments from LOGAN1 image

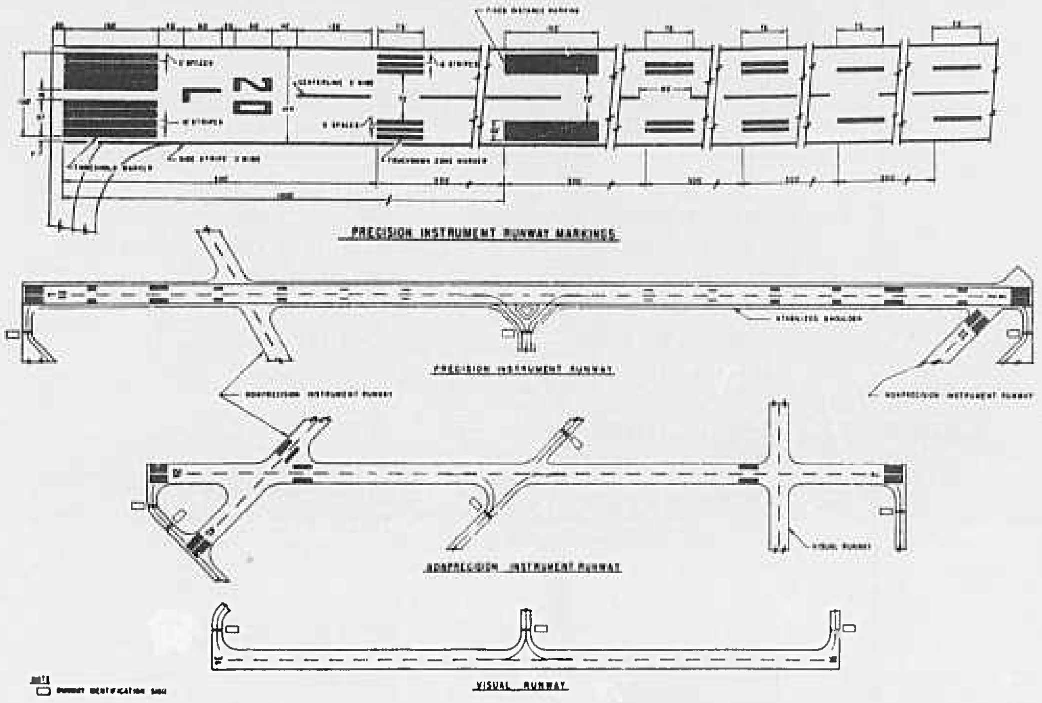

PRECISION INSTRUMENT RUNWAY MARKINGS
PRECISION INSTRUMENT RUNWAY
NONPRECISION INSTRUMENT RUNWAY
VISUAL RUNWAY

Figure 5: Standard Runway Markings

aircraft we can expect to see; large-wing span aircraft require precision instrument runways, for example. For a complete discussion on runway and taxiway markings, see the ICAO Annex 14 and [7]. Our description relates to the FAA specifications, which are generally similar in function and form to the international standards; where differences occur, they are not sufficiently great to cause confusion [6].

In summary, a runway is modeled as a long rectangle containing various distinguishing markings. We first hypothesize their presence and then verify them as runways, as depicted in figure 6.

Our hypothesis formation process consists of the following steps:

1. *Detect Lines Segments and Anti-parallelism*: We apply one of our edge detection techniques [1,3] to obtain the line *segments* corresponding the intensity edges in the image. We next detect *anti-parallels* (pairs of segments of opposing contrast) from the line segments.

2. *Estimate Runway Directions and Widths*: We compute the length weighted histogram of apar directions and use the peaks to select one or more runway orientations. To estimate runway width, we compute a length weighted histogram of the apar widths (the distance between the anti-parallel pair of segments) for each of the selected estimates of runway orientation. The peaks in the histogram correspond to sets of *apars* having similar widths.

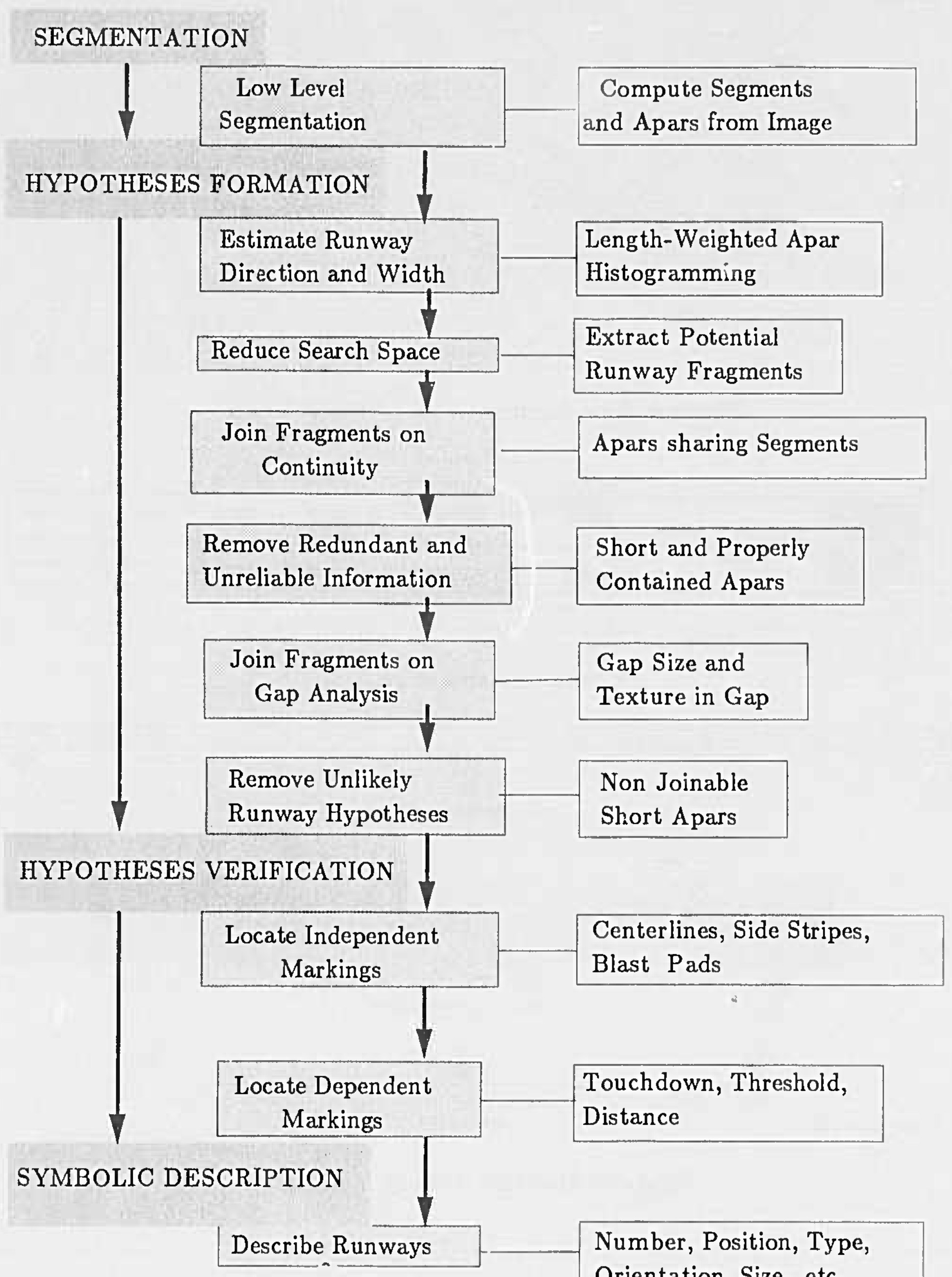

SEGMENTATION
Low Level Segmentation
Compute Segments and Apars from Image
HYPOTHESES FORMATION
Estimate Runway Direction and Width
Length-Weighted Apar Histogramming
Reduce Search Space
Extract Potential Runway Fragments
Join Fragments on Continuity
Apars sharing Segments
Remove Redundant and Unreliable Information
Short and Properly Contained Apars
Join Fragments on Gap Analysis
Gap Size and Texture in Gap
Remove Unlikely Runway Hypotheses
Non Joinable Short Apars
HYPOTHESES VERIFICATION
Locate Independent Markings
Centerlines, Side Stripes, Blast Pads
Locate Dependent Markings
Touchdown, Threshold, Distance
SYMBOLIC DESCRIPTION
Describe Runways
Number, Position, Type, Orientation, Size, etc.

Figure 6: Block Diagram of Runway Detection System

3. *Join Apars based on continuity*: Apars with a selected orientation and having similar widths are joined if they share a common segment, are collinear and have the same color (brighter or darker than surround) to form a new longer apar.

4. *Remove Apars with Low Aspect Ratio*: Apars with an aspect ratio of less than 1 are considered unreliable and thus, removed from the set.

5. *Remove Contained Apars*: Apars which are inside other apars are removed to simplify the joining process. The evidence provided by the surrounding apar is judged to encompass the evidence given by the smaller, thinner apar.

6. *Join Collinear Apars:* After joining on the basis of continuity, many runway fragments remain fragmented due to occlusion and other factors already mentioned previously. In this step we join collinear apars of similar width if the gap between them is small compared to the lengths of the candidate apars or, if the *segments* between the candidate apars show strong evidence of features oriented in the direction of the apars.

The resulting apars over a given aspect ratio are the runway hypothesis. Although this process is liberal in its decisions to join candidate apars, 3-D information (from stereo) should help in some cases to verify a smooth and flat surface. Our current system, however does not make use of 3-D information.

To verify our hypotheses as runways, we use airport engineering knowledge [6], and specifically the characteristics of the standard and optional runway markings. Some of these are shown in figure 5.

Usually, it is possible to give the image resolution *a-priori* to help locate specific markings. However, our current implementation also computes an estimate of the image resolution as it gathers evidence, for example, from the knowledge of runway widths and the widths of the formed hypotheses. The image resolution is used as a guide to help locate the specific runway markings from the locations of previously detected markings, and to help predict the location of markings otherwise not visible or undetected. If the image resolution is available *a-priori*, comparison to the estimate computed also helps to indicate that the evidence obtained is consistent with that we expect to find.

The runway verification process attempts to locate, in some cases independently (see below), the following evidence:

1. *Runway Centerlines*: We detect *bright* apars of given length and width near the center of the hypothesized runway. These apars should be interspaced by a common distance.

2. *Side Stripe Markings*: We look along the sides of the hypothesized runways for evidence of long thin *bright* apars. These correspond to side stripes which delineate the runway *landing* surface.

3. *Threshold Markings*: All runways are assumed to have a pair of threshold marks at each end which are at least partially visible. These marks are the begin and end points of the runway, even though the paved surface may extend beyond these marks. If only one of the pair or parts of the threshold marks are available, the position of the mark or matching mark is hypothesized from available apar or segment information.

4. *Touchdown Markings*: The threshold marks (above) are assumed to delimit the runway surface. If threshold marks are not available, it becomes difficult to locate touchdown and distance markings. Otherwise, from the position of the threshold marks, we can determine where the touchdown marks should be. We search the immediate area for a *bright* apar of given length and width. If we cannot find a bright apar to match the estimated length and width, we look for a *dark* apar of similar proportions.

5. *Large Distance Markings*: Runways usually have distance markings of various sizes at equally spaced intervals. The first set of distance marks is large and is usually easy to find. From the position of the large distance markings, we further refine our estimate of the resolution of the image for more localized searches.

6. *Small Distance Markings*: From the position of the large distance marks, we compute the approximate location of the four sets of smaller distance markings and look for suitable apars near these locations.

7. *Blast Pad Markings*: These are optional markings located at the end of the runways, and beyond the threshold markings. The are characterized by their arrangement into a "herringbone pattern". Although thin, when there is sufficient resolution in the image, these are reliable features that form thin *bright* apars located at a known angle with respect to the runways.

3 DETAILS OF THE METHOD

We will now give further details of our method, and its implementation.

3.1 Formation of Runway Hypothesis

3.1.1 Detection of Line Segments and Anti-parallels

We use the USC "LINEAR" line detection system to obtain line *segments* and *apars*. Edge detection is performed by applying either the Nevatia-Babu [1] or the Marr-Hildreth [5] edge detectors to the image. Edges are then thinned and linked to form continuous curves. These curves are approximated by piecewise linear segments. Each linear segment is described by its length, orientation, contrast, and position of its end points. Additionally we also know if a segment connects to another segment at either end. Figure 4 shows the 8262 line segments computed from the image of LOGAN's Logan International Airport (LOGAN1) shown previously in figure 2. The apars are computed from the segments by specifying the minimum (in our examples, 1 pixel) and maximum (60 pixels) distance between the anti-parallel pairs of segments. The range will be known in practice if the altitude of the aircraft and image resolution are known. Figure 7 shows the center axis lines of the *apars* computed from the segments shown in figure 4. For details on the computation of apars, see [1]. The *apars* are described by their length, orientation, end points, width and color (brighter or darker than surround). We also know if apars are connected to other apars at either end.

3.1.2 Hypothesis of Runway Orientation and Width

In this step we attempt to estimate the direction and width of the runways in the image. We first estimate the direc-

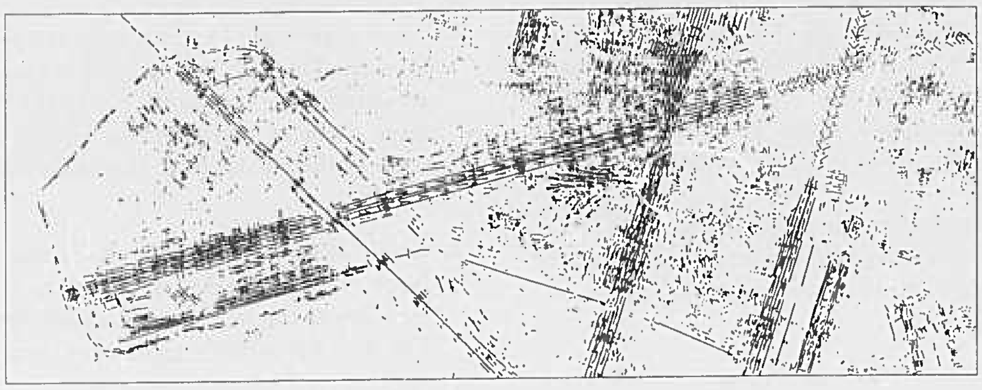


Figure 7: Anti-parallels from segments in LOGAN1 image

tion of the runways by computing a length weighted histogram of the orientation of the detected apars . The resulting histogram for the LOGAN1 image is shown in figure 8. The three peaks detected denote the dominant orientations, even when the runways are only partially visible. In our example, the peaks are at 43°, 63° and 123°, with 0° pointing south.

To obtain an estimate of the runway widths in the image we compute a length weighted histogram of the apar widths for those apars oriented in the estimated runway directions. The resulting histogram, shown in figure 9, shows three groups of apars. Typically the group of wider apars (between 30 and 40 pixels in our example) contains runway and shoulder fragments. The middle group (between 10 and 25 pixels) contain taxiways and service roads, and in some cases, narrow shoulders. The third group (between 1 and 6 pixels), contains the surface markings.

Although the above histogramming techniques have been sufficient to estimate direction and width, variations of the technique can be implemented. For example, in estimating runway direction, we can use the segments instead of the apars, which in greater numbers, would contribute more heavily to the histogram. In estimating widths, we can compute one histogram for each of the selected orientations to obtain better definition of apar groups.

3.1.3 Construct a Set of Runway Fragments

In this step we reduce the search space for runway fragments using our estimates of runway direction and width. We extract form the set of *apars* those in the selected directions and belonging to the width group for runways. In our LOGAN1 example, we construct three sets of runway fragments, one for each of the three selected runway orientations (43°, 63°, and 123°) allowing for a tolerance of 5° on both sides of the histogram peaks. The sets contain the *apars* in the runway width group (in our example, between 25 and 55 pixels). The resulting three sets for the LOGAN1 example are shown in figure 10. The apars are shown as rectangles to show their width. These *apars* represent a

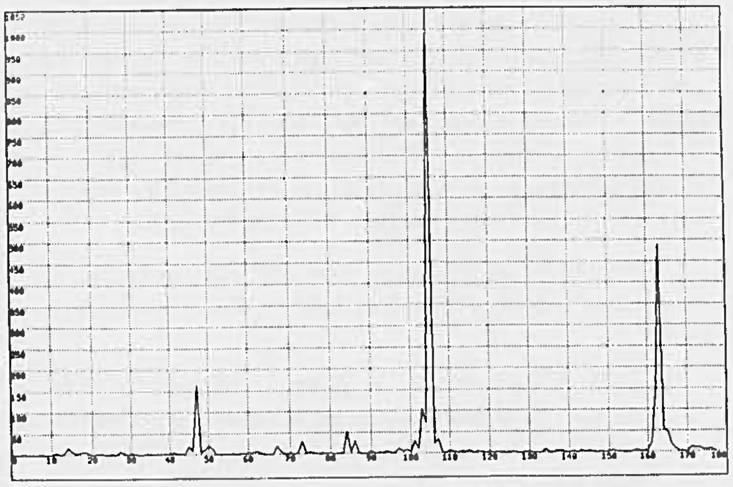


Figure 8: Length Weighted Histogram of Apar Orientations

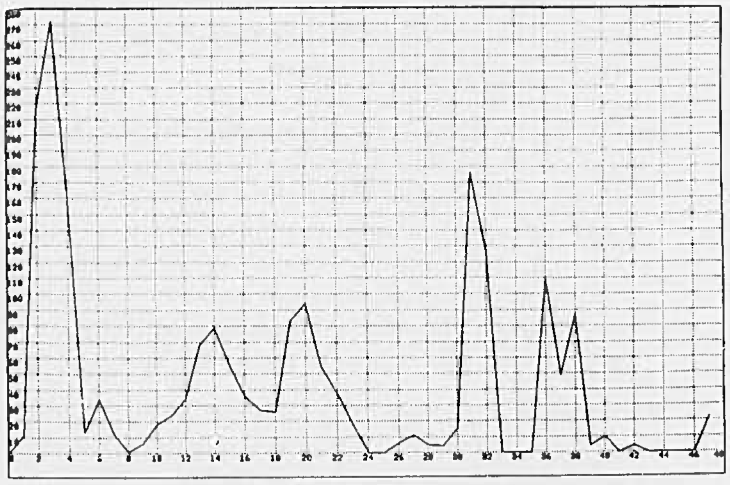


Figure 9: Length Weighted Histogram of Apar Widths

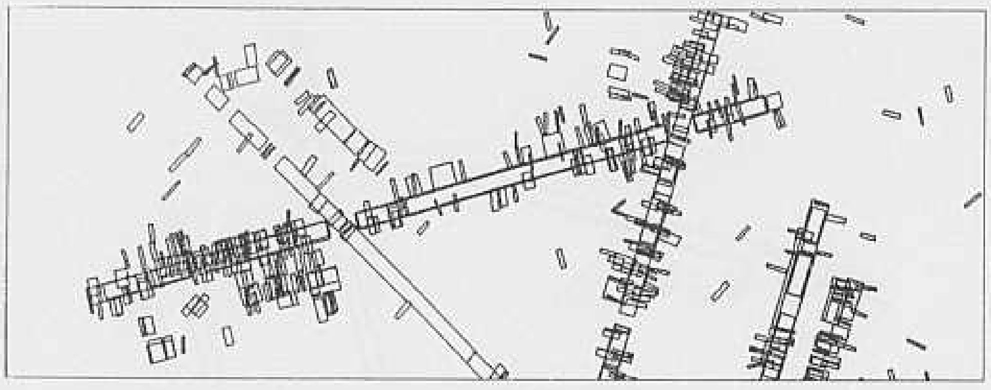


Figure 10: Apars representing initial set of runway fragments

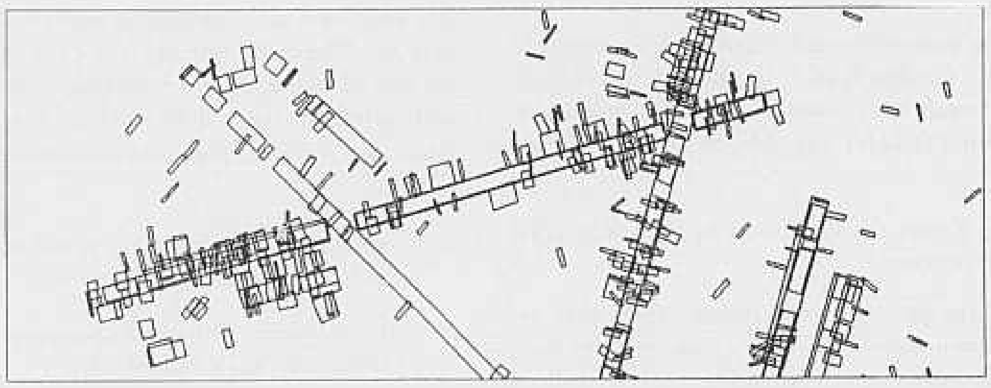


Figure 11: Apars Joined on the basis of Boundary Continuity

strong set of evidence of the presence of airport runways (and other long linear structures) in the image. Compare the original set of *apars* to those shown in figure 8. The search space was reduced, in this example, by 94%, from 9,498 in the entire set to 518 apars.

3.1.4 Joining Apars on the Basis of Continuity

Apars representing linear structures are usually broken due to a variety of factors, such as oil spots and markings, noise in the image and inadequacies in the low-level processes.

Additionally, some of the breaks are due to real structures in the image. Consider for example where taxiways join runways. One one of the boundaries of the runway is seen as a continuous linear structure while the other boundary is broken at the junctions. Typically the segments representing the continuous boundary will form *apars* with several segments corresponding to the broken boundary, leading to a sequence of collinear apars having the same width. Typically these fragments will also have the same color.

In this step we join the apars that share segments, are collinear, and have the same color. In our examples we have allowed a 5° tolerance in collinearity and 5 pixels tolerance in width. The resulting longer *apar* must have an orientation that is compatible with the estimated direction of the runway within a small tolerance (5°). The result of joining apars in this manner is shown in figure 11.

3.1.5 Removing Redundant Information

In some cases, as in our LOGAN1 example, there is sufficient resolution in the image for the edge detector to be able to resolve portions the white side stripes that bound the landing surfaces of the runways. In these cases the outside boundaries of the white side stripes result in apars that contain apars resulting from the inside boundaries of the same white side stripes. We consider *properly contained* apars to be additional and redundant information that can be ignored and removed from the set. Apars that overlap however are preserved.

Note that it may appear that apars formed by the boundaries of the shoulders would contain apars formed by the runway boundaries, and therefore may be removed from the set, but we have found this not to be the case. In general the shoulder boundaries are not as homogeneous as the runway boundaries and tend to be quite broken. Once runway apars have been joined on continuity alone, although thinner than shoulder apars, they tend to be considerably longer, and therefore, not contained in wider shoulder apars.

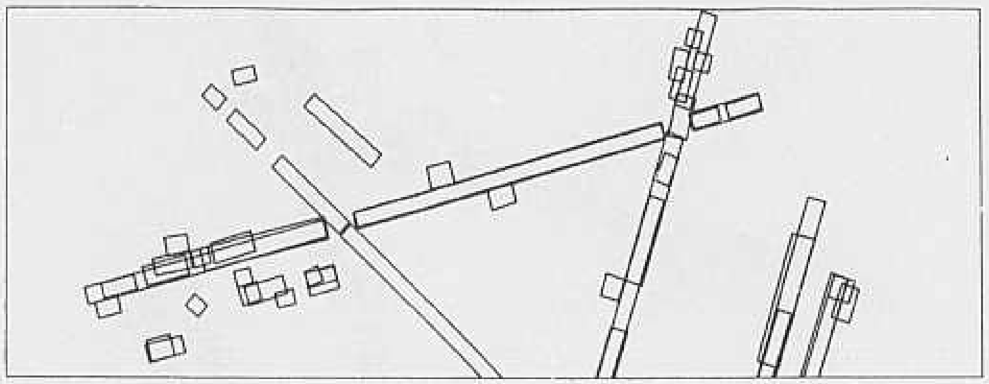


Figure 12: Apars filtered on containment and aspect ratio

In this step we also remove apars having an aspect ratio of width to length smaller than 1, as they are considered unreliable. The result of removal of contained apars and small apars in our LOGAN1 example, is shown in figure 12.

3.1.6 Joining Apars on Collinearity and Analysis of Gap Texture

At this stage of the process, many runway apars may remain broken due to noise and occlusion. Consider for example where two runways cross or when there are aircraft on the runways. Next we discuss our criteria to join collinear apars that have similar widths on the basis of examining the gap between the fragments. In general, this process is quite liberal in the analysis of the information in the gaps. For instance, if the gap contains mostly segments that are oriented in the direction of the apars, we join them. If the gap contain mostly segments oriented at an angle consistent with the angles allowed between crossing runways then we join them. However, as in our JFK1 example shown in figure 34, repair work, changes in surface material, signs of heavy use, oil spots and tire tread marks, can result in basically random arrangements of segments (texture) in the gaps. Thus, to allow for these, we determine our decision to Accept or reject the information in the gap to join two candidate apars as a function of the lengths of the apar candidates and the size of the gap: Texture in the gap is less important for small gaps than for large gaps; angle tolerances are tighter for short candidates than for long candidates. A more precise way to implement these decisions would be to use 3-D information to determine if the surface is smooth and flat in 3-D. Such information would be available from stereo.

The joining process is currently implemented as follows:

First, we order the sets (one for each selected runway direction) by apar length. We then try to join the apars in each set, beginning with the longest apar and cycling through the apars in decreasing order of length. For each of these apars, we look for candidate apars off both ends of this apar for a set of candidate apars to extend the original apar to. The candidate apars are sorted by distance from the end of the apar we are seeking to extend. We consider each candidate apar individually. The decision to join a given apar to a candidate apar is subject to the following criteria:

1. *Width:* The apars to be joined must have the same width (with a 5 pixel tolerance).

2. *Parallelism:* The candidate apars must be parallel (with a $5°$ angle tolerance).

3. *Collinearity:* The candidate apars must be collinear (within 1 pixel).

4. *Orientation Consistency:* If joined, the orientation of the resulting apar must be consistent (within $5°$) with the estimated runway orientation (from the length weighted orientation histogram).

5. *Inter-Apar Gap:* The length of the gap must be smaller than the sum of the lengths of the candidate apars.

6. *Segment Texture Check:* If most of the segments contained in the gap between the two apars are oriented in the direction of the apars, then the apars is joined. To determine this, we compute a length weighted histogram of the segments in the gap, and select the peak orientation. This orientation must be consistent with the direction of the hypothetical runway.

   If most of the segments in the gap have a dominant orientation, but different from the orientation of the apar candidates, the angle difference is compared to the allowed angles between crossing runways.

   In general, if the gap is very small (twice the hypothetical runway width), we ignore the texture check and join the apars.

This joining process is continued until the process has stabilized and no further joins are possible. For our LOGAN1 example, the result of this process is shown in figure 13.

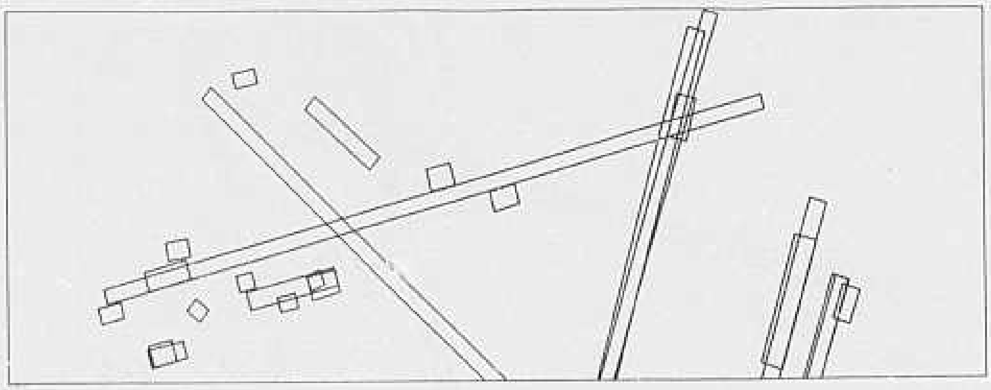


Figure 13: Apars Joined on Segment Texture and Gap analysis

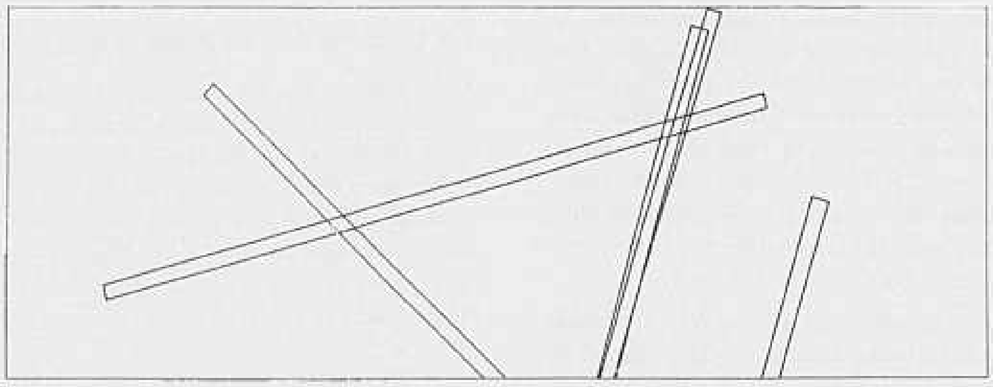


Figure 14: Runway Hypothesis

3.1.7 Final Runway Hypotheses

At the end of the joining process, short apars are removed from the sets if they have an aspect ratio smaller than 20:1 with respect to a runway aspect ratio. This will preserve those apars possibly representing partially visible runways. The resulting apars constitute the instances of the shapes found in the image that match our geometric model for airport runways. These are shown in figure 14 for our LOGAN1 example.

3.2 Runway Verification

Runway hypotheses represent instances of the shapes found in the image that fit our runway model. Runway verification attempts to validate the hypotheses made by locating the markings associated with runways. Our technique assumes that these markings are at least partially visible and resolved by the low level segmentation technique used.

In general, to find the markings we first look for thin bright apars. If necessary we also look at the segments. The 1817 thin apars in our LOGAN1 example are shown in figure 15. Detection of markings is described in detail below. These are:

- Standard Markings.
  - *Runway Centerlines.*
  - *Side Stripes.*
  - *Threshold Marks.*
  - *Touchdown Marks.*
  - *Distance Marks.*
- Standard Markings (optional).
  - *Blast Pad Marks.*

The visibility of runway markings is primarily determined by the resolution of the image and by the local contrast:

- *Image Resolution*: Depending on the low level segmentation used, the (usually) white markings on runways and the (usually) yellow markings on taxiways can be resolved. In our examples, the side stripes in LOGAN1, for example, are about 5 pixels wide, allowing us to detect them easily.
- *Surface Material*: White markings on a dark asphalt surface are quite visible if the landing surface and

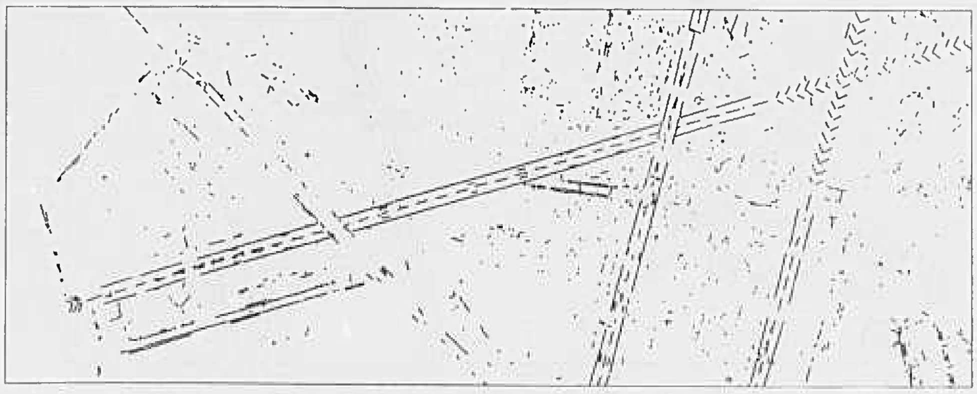


Figure 15: Thin anti-parallels in LOGAN1

the shoulder are of the same asphalt material. In some cases the shoulder is of a lighter asphalt material and only one boundary of the markings (side stripes) has sufficient contrast. In the former case, the white markings denote the runway boundaries, and sufficient resolution is required to detect them. In the latter case, the boundary between the landing surface and the shoulder may be detected easily, even if the white markings can not be resolved.

More durable materials, such as concrete are bright and perhaps make it more difficult to detect the white markings. In many cases however, for cost reasons, the shoulders of concrete runways are of a darker material (asphalt), and provide a high contrast between runway and shoulder. A future extension to our technique will attempt to locate markings on concrete runways by either applying a specialized edge detector to the weak and low contrast markings, or by examining the intensity data directly.

- *Usage and Upkeep*: In most cases, tire tread marks, oil spots and exhaust fumes obscure some of the markings. On the other hand, tire tread marks form quite visible and high contrasting dark regions in the center of concrete runways, and can be used for verification purposes, even if the markings are not visible or detectable. Exhaust fumes obscure the markings at the end of runways. In this cases our current technique relies on markings detected elsewhere to predict the presence of obscured markings. Again, a dedicated edge detector or examination of the intensity data should help.

- *Non Standard Markings*: Figure 5 showed the standard markings for three types of runways in commercial airports. Our JFK1 example on the other hand shows that the left (concrete) runway contains non standard markings (both in size and in distance). These are not detected by the current implementation. As an extension to our implementation we plan to characterize and implement the detection of these markings.

3.2.1 Detection of Runway Centerlines

According to runway marking standards, the centerlines are supposed to be 3 feet wide and 120 feet long, spaced every 80 feet along the landing surface of the runway. To detect centerlines we look in the middle of the hypothesized runway for bright apars which are less than 5 feet wide and between 40 and 140 feet long. These also must be oriented in the direction of the hypothesized runway. The centerlines located for our LOGAN1 example are shown in figure 16.

In the current implementation, we are only looking for apars of appropriate length down the center of the hypothesized runways, regardless of the 80 feet separation constraint. This allows detection of broken or incomplete individual markings due to exhaust burns, tread marks, etc. For the same reason we also look for individual segments (that do not form thin apars) down the middle of the runway. This however may result in detection of some "stray" segments corresponding to repair work and other features on the landing surface.

3.2.2 Detection of Side Stripe Markings

Side stripes bound the sides of the landing surface of runways. Side stripes are at least 3 feet wide. If sufficient resolution is available and there is sufficient contrast, we are able to detect at least one of the boundaries of the stripes along the runway. If both boundaries are detected, side stripes are detected as thin bright apars (see figure 16), at or near the boundaries of our runway hypotheses, and oriented parallel to the estimated runway direction. These thin apars are often broken mostly due to lack of resolution, and we do not attempt to join them. We however require that they be bright, collinear and that they have a consistent width.

The estimates for runway orientation may be off a few degrees from the actual runway direction in the image. This is due to minor angular adjustments made to the apars resulting from the joining processes when the hypotheses are formed. To allow for this variations, we look for side stripe apars in a window equivalent to the length of the

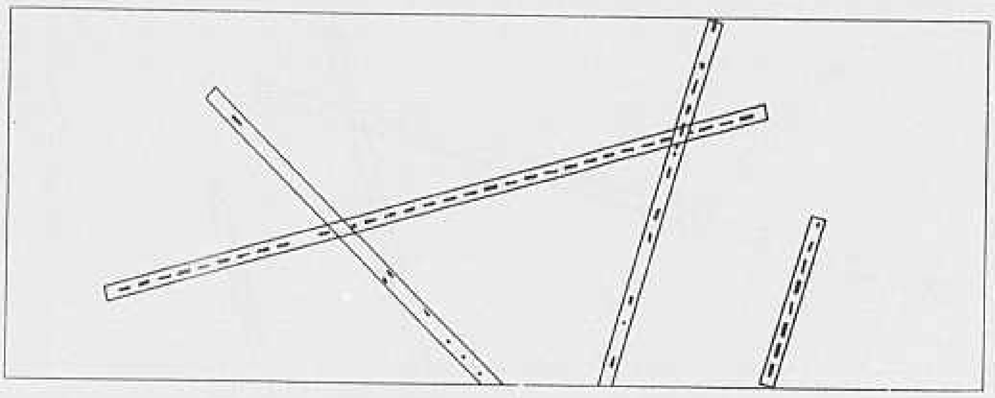


Figure 16: Centerlines detected in LOGAN1

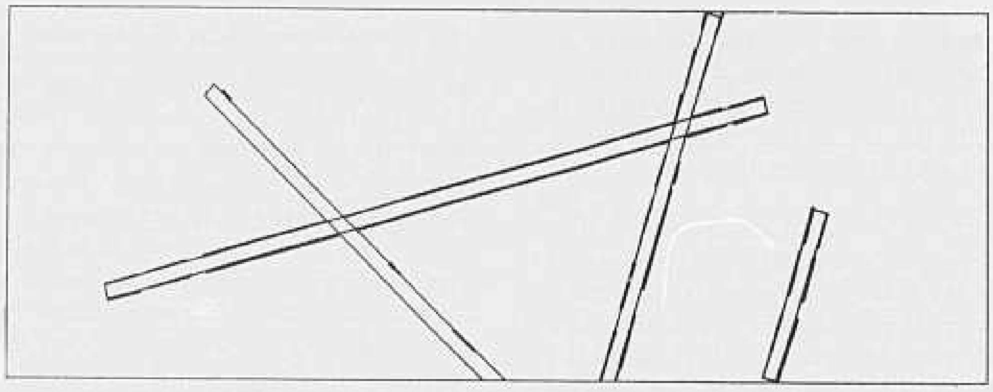


Figure 17: Side stripe Markings in LOGAN1

hypothesized runway, and having a width equivalent to 15 feet. That is, we allow an total error margin of 6 feet on both sides of the hypothesized runway boundary. The apars corresponding to side stripes in our LOGAN1 example are shown in figure 17. Note that one of the overlapping (competing) hypothesis (see figure 14) can be discarded because it has only a few centerlines compared to those in the hypothesis that remains valid.

3.2.3 Threshold Mark Detection

The threshold marks consist of a pair of four closely painted 12 foot white lines, 3 feet apart, and 150 feet long, separated by a dark rectangular zone 16 feet wide. The distance between these markings and the side stripes is a dark zone 7 feet wide.

In our model, all runways are assumed to have a pair of threshold marks at each end of the runway (see figure 5). These are probably the most important set of markings that can be used to verify a hypotheses as a runway; they give pilots the position of the start and end of the runway. Often, these marks are partially worn away by exhaust fumes due to their position so we expect our search to look for partial markings. However, if we cannot find evidence of these marks in any form, we discard the runway hypothesis.

The amount of segment and apar information detected, outside of contrast, depends on the image resolution. If high resolution is available (higher than that of our examples), then between the two side stripes we would have 20 parallel line segments, contributing to about 50 apars. Even with partial information the marks can be identified easily. At low resolution (less than that of our examples) we would have two segments corresponding to the outside boundaries of the side stripes, and one bright apar as wide as the runway. High contrast would be required to detect this apar.

At the resolution in our examples it is difficult to resolve the individual lines, and the threshold marks appear as white rectangles 150 feet long and 57 feet wide, separated by a dark zone 16 feet wide. This results in two bright 25 feet wide apars for each mark and a 16 feet wide dark apar between them. In our search first look for the bright apars. These apars must be oriented in the direction of the runway (within a 5° tolerance).

We expect to find a pair of apars which fit this description, however, often there is only be one apar found. In this case, we can hypothesize the position of the missing mark. The missing mark will have the width and length of a threshold mark in the FAA model with position and orientation of the mark determined by the position and orientation of the apar which was found. From the position

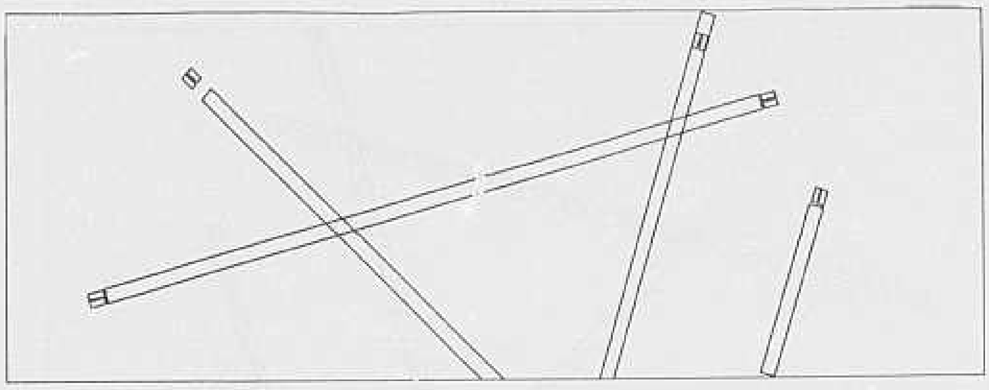


Figure 18: Hypothesized sets of Threshold Marks

of the found apar, and the knowledge that the threshold marks have 16 feet between them, we can accurately determine the position of the mark. Using this information, we can now go back to the line segment information and look for a line segments to support our hypothesis.

In some cases neither of the bright apars may be visible or detected. The next available and reliable feature is a *dark* apar in the middle of the runway (collinear with the centerlines). The dark apar must meet the length and orientation constraints for the dark zone between the thres. old marks; it must be 16 feet wide and no longer than 150 feet allowing a few feet of tolerance (between 10 and 19 feet wide). From the position and orientation of this dark apar, we can make accurate predictions as to the position and orientation of the two threshold marks. As mentioned before, we also search for evidence in the set of line segments to support of this hypothesis. In our examples this has been sufficient but using the pixel data may help in some cases.

Although we know that threshold marks are located at the end of the runways, the hypotheses we form from the apars may not extend to the ends of the underlying runways. In other cases the side stripes are extended beyond the threshold marks, causing the threshold marks to be "inside" the hypothseized runway. Our search window therefore collects evidence inside a window that extends from inside the hypothesized runway and in the direction of the runway, beyond the hypothesized runway end. The window is also wider than the width of the hypothesized runway.

We expect to find more than one configuration of apars and/or line segments that potentially represent the threshold marks. Since runway markings are constrained by position and size, we test all potential pairs of threshold marks against other markings to select the pair that assures consistency Figure 18 shows the threshold marks for our LOGAN1 example. Once a set of markings is found the runway hypothesis can then be updated.

3.2.4 Touchdown Mark Detection

The touchdown marks consist of three 75 feet long 6 feet wide stripes, 5 feet apart. At the resolution in our examples, the individual stripes can not be resolved, and are detected having a width of 28 feet. The are located on each side of the runway, with 72 feet between them. They are located 340 feet down the runway from the threshold marks. In our current implementation we look for two bright apars and or a dark apar in the approximate position predicted for the touchdown marks, and subject to the orientation constraint. In neither of these are found we can also look for line segments although this additional search has not been implemented yet. The detected touchdown marks for our example are shown in figure 19.

3.2.5 Distance Marking Detection

Runways have a series of distance markings extending from the touchdown marks, starting at 500 feet from the touchdown marks, and located 500 feet apart. The first pair (large fixed distance markings, in figure 5) consists of two 150 feet by 30 feet stripes, separated 72 feet. The rest of the distance markings are similar to the touchdown markings, except that the first two (after the large first pair) consist of two 75 feet by 6 feet stripes and the subsequent ones consist of only one 75 feet by 6 feet stripe. The distance between the two marks in each pair is the same, 72 feet. At the resolution in our examples, the first two stripes in each mark can not be resolved by the low level segmentation technique used, and are detected as a single bright apar 17 feet wide.

We look for the first (large) pair of distance marks first. For this we rely on the position of the threshold marks to predict the approximate positions for these large distance markings. We look for a bright apar oriented in the direction of the runway which is at least 100 feet long and 20 feet wide, subject to the particular size constraints (it cannot be more than 150 feet long or 30 feet wide ±5 feet). We also allow a 5° tolerance in angles. As before, we choose among several candidates if necessary based on proximity to the predicted position.

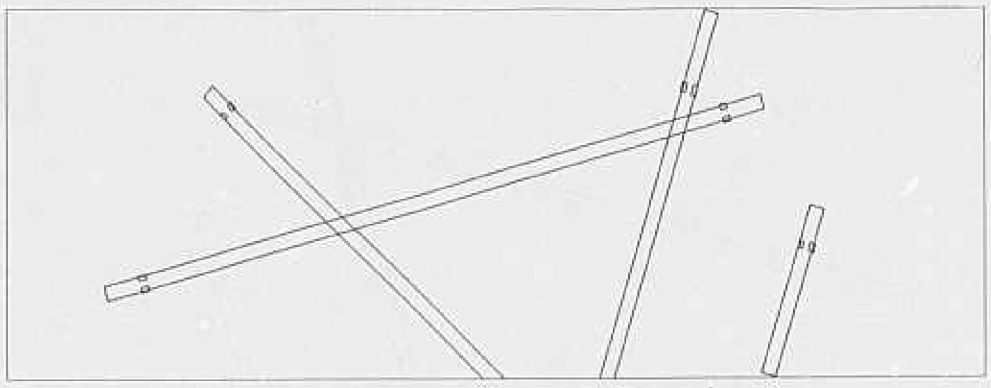


Figure 19: Touchdown Marks Detected

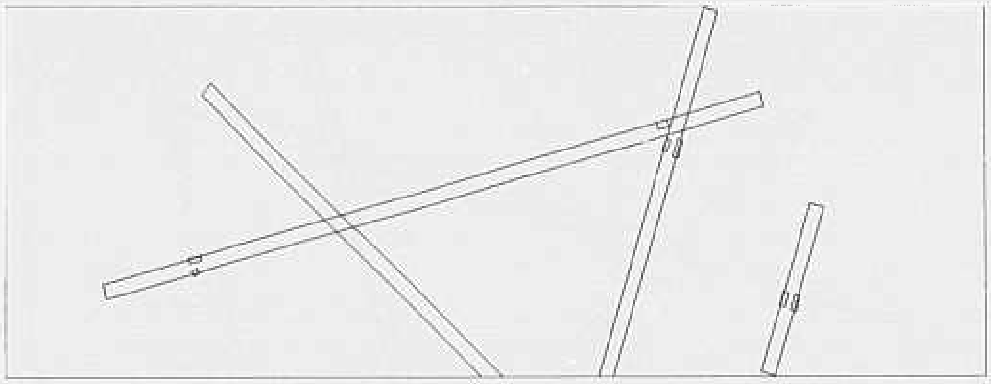


Figure 20: Large Fixed Distance Markings in LOGAN1

Once we find large distance marks, we further refine our estimate of image resolution. Recall that the initial estimate of image resolution is based on *a-priori* knowledge of the widths of commercial runways compared to the hypothesized runway widths. However, it is possible that narrow shoulders be included in our initial runway hypotheses. These refinements are important to locate small and more difficult to detect markings.

Locating the other small distance markings proceeds in a similar manner. We estimate their position from the large distance marks (if these are available, otherwise we use the position of the threshold marks) and do a search in the area for an apar of the desired characteristics. The distance markings found for our LOGAN1 example are shown in figures 20 and 21.

3.3 Blast Pad Mark Detection

Blast pad markings are optionally located at the ends of runways. They consist of pairs of white lines oriented at 45° angles with respect to the runways, and meet at the runway central axis. Also they do not extend beyond the width of the runway landing surface. The separation between these pairs of lines varies thus, we detect them by looking for thin bright apars in the proper configuration. The blast pad markings detected for LOGAN1 are shown in figure 22.

3.4 Runway Mark Set Selection

Our search process for runway marks is chiefly based on finding the threshold marks at the end of the runway. We have used all the potential threshold marks as guide to locate other markings, and we have labeled the evidence found accordingly. In this stage we evaluate these labelings to determine the best overall set of markings.

In general we expect that the candidate threshold marks that generated the largest set of consistent markings be the best set. To select the best set then we implemented a weighted function of markings, assigning arbitrarily twice more weight to the markings we consider more important, the touchdown markings and the large fixed distance markings. All other markings have the same weight. Note that this simple function is adequate since an incorrectly labeled set of threshold marks would generate incorrect predictions for the rest of the markings, and therefore will generate a small set. The resulting set of markings for our LOGAN1 example is shown in figure 23.

4 More Results and Comments

We have tested our method on several images of major commercial airports. In our discussion we showed results on a portion (LOGAN1) of Logan International Airport in LOGAN. In this section we present results for another portion

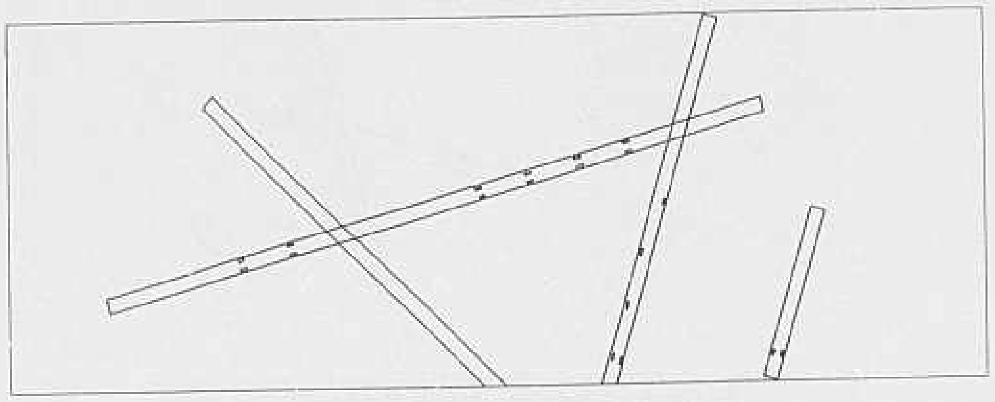


Figure 21: Distance Markings in LOGAN1

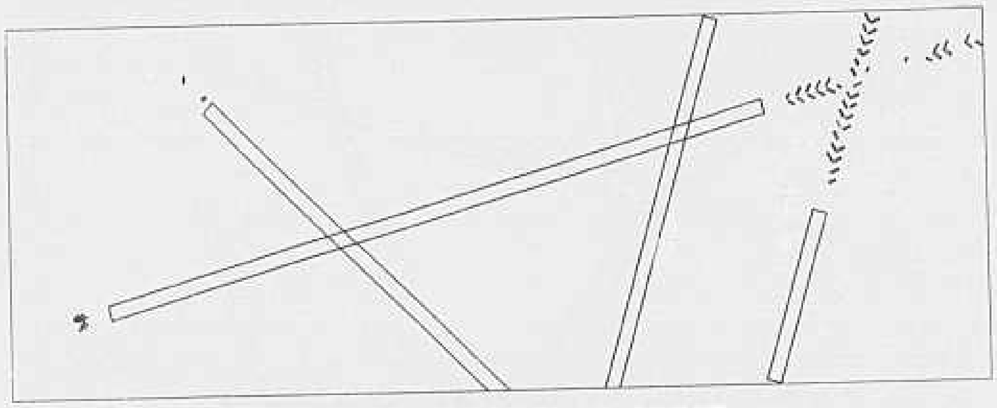


Figure 22: Blast Pad Markings in LOGAN

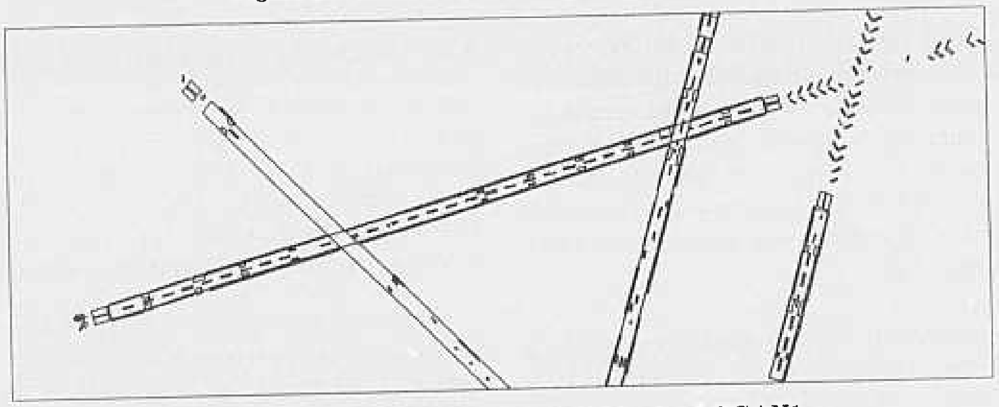


Figure 23: Selected Set of Runway Markings for LOGAN1

(LOGAN2) of the same airport. The runways at Logan Airport consist of dark asphalt, well maintained surfaces and markings, while JFK present a wide variety of problems. We therefore selected two portions (JFK1 and JFK2) of this airport as our second example. The level of complexity of most major commercial airports lies between our two examples.

Figure 24 shows another portion (LOGAN2) of Logan International Airport (2300×1200 resolution). The line segments computed from this image are shown in figure 25. The 22,691 apars computed from the segments are shown in figure 26. Note the complexity and size of the original search space. The estimates for runway directions are chosen to be the three peaks form the length weighted histogram of the orientation of the apars, shown in figure 27. The runway width estimates are obtained form the length weighted width histogram of the apars in the selected directions as described above for our LOGAN1 example. The reduced search space and apars representing the initial set of runway fragments is shown in figure figure 28.

In this example, the shoulder of one of the runways has a width similar to that of the runways, which result in many "non-runway" apars being formed. Figure 29 shows runway

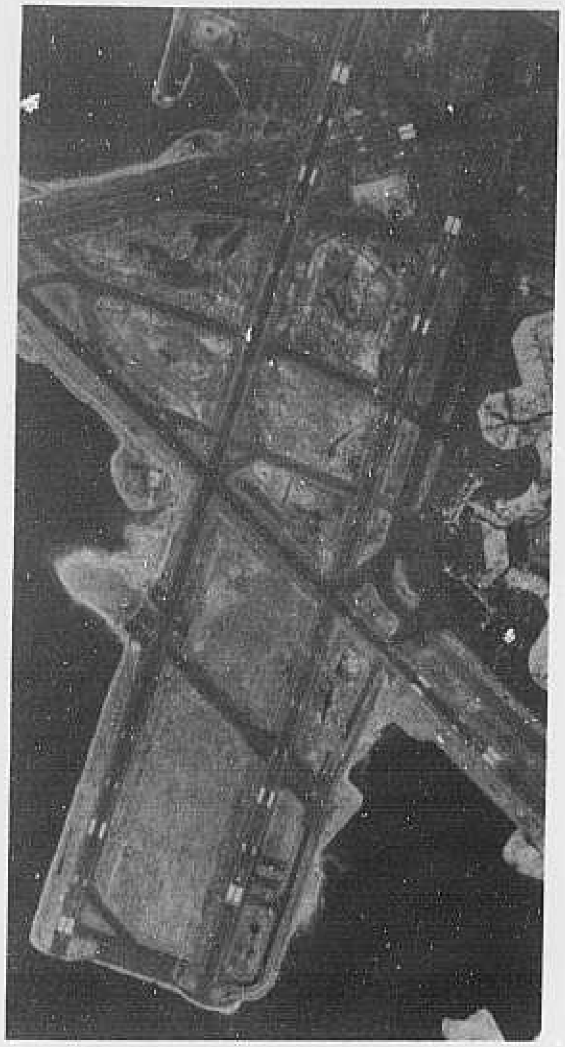


Figure 24: 2300x1200 pixel LOGAN2 Image

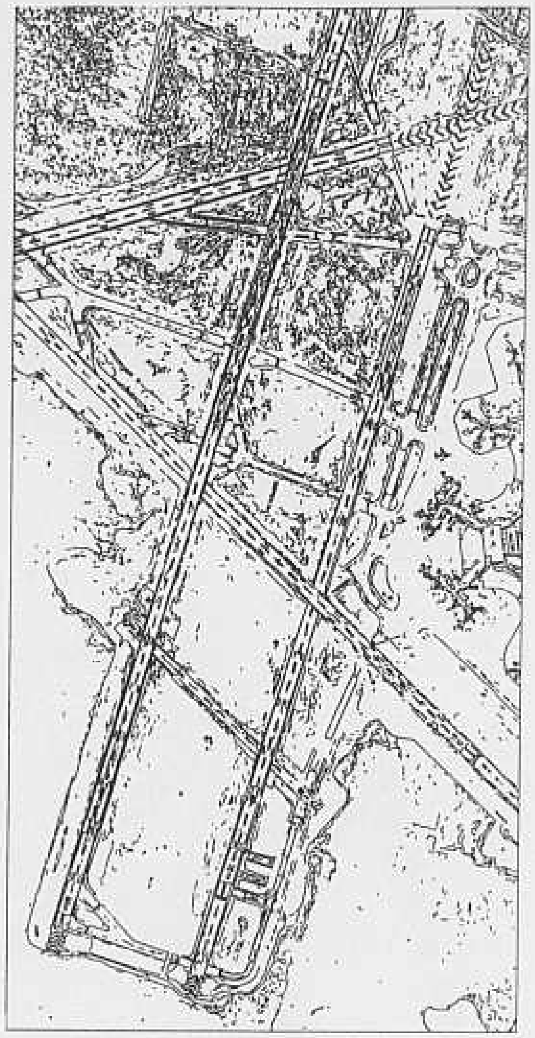


Figure 25: Line Segments from LOGAN2 Image

apar fragments joined on boundary continuity. Figure 30 shows the apars remaining after removal of the apars with an aspect ratio of less than one, and removal of properly contained apars. The apars are then joined on the analysis of the gaps are shown in figure 31. The apars thresholded on aspect ratio to give the runway hypothesis are shown in figure 32. Note the overlapping hypotheses due to a runway and its shoulder. These are disambiguated in the verification step that follows. Figure 33 shows the results of the verification process. The hypothesized runway which was actually the apar formed by the shoulder of a runway was easily eliminated in the verification process.

Our next example presents two portions (JFK1 and JFK2) of John F. Kennedy International Airport in New York. Figure 34 shows a portion (JFK1:1500 × 2600 resolution) of John F. Kennedy International Airport in New York. The partially visible apparent runways have no discernable markings on them. The complete runway running across the image shows increasing amounts of repair work, of a different material than that of the original surface. The darker material, however, makes some of the markings more visible. On the left side of the runway, the end of the runway becomes less wide as it turns into a taxiway. The accurate detection of the runway end thus depends on being able to locate the threshold markings. As shown below, we were able to locate them.

The line segments computed from the JFK1 image are shown in figure 35. The 17,766 apars computed from these segments are shown in figure 36. The estimates for runway directions are chosen to be the two peaks form the length weighted histogram of the orientation of the apars, shown in figure 37. The runway width estimates are obtained form the length weighted width histogram of the apars in the selected directions as described above. The reduced search space and apars representing the initial set of runway fragments is shown in figure 38. Figure 39 shows runway apar fragments joined on boundary continuity. Figure 40 shows the apars remaining after removal of the apars with an aspect ratio of less than one and contained apars. The apars are then joined on the basis of analysis of the gaps between fragments are shown in figure 41. The thresholded apars on aspect ratio give the runway hypothesis, shown in figure 42. Figure 43 shows the results of the verification process. Note that we are able to locate some of the centerlines due to the darker material used in the repair work.

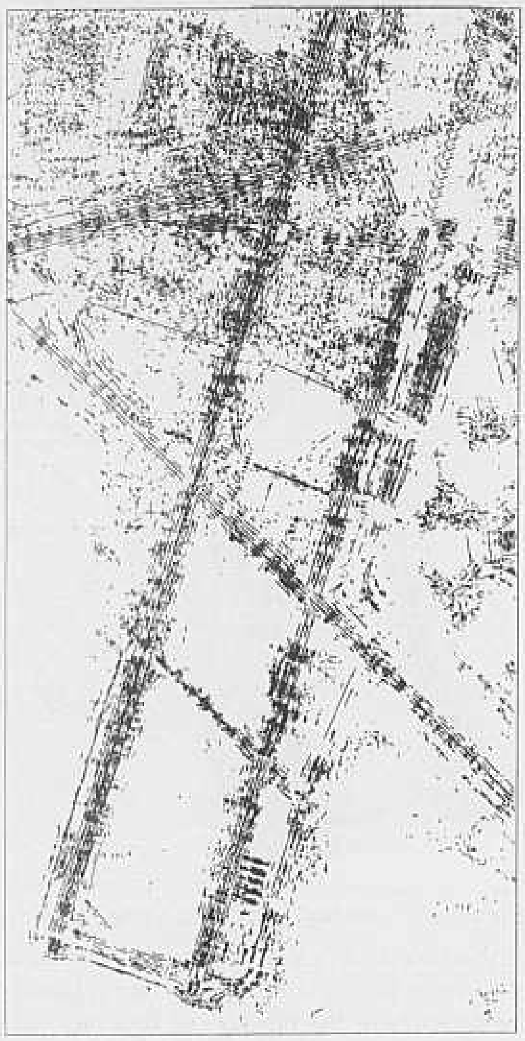


Figure 26: Anti-parallels from line segments in LOGAN2

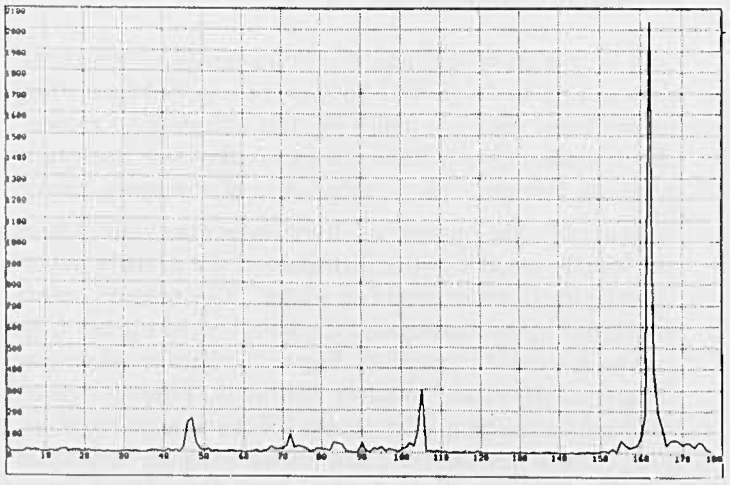


Figure 27: Length weighted apar orientation histogram for LOGAN2

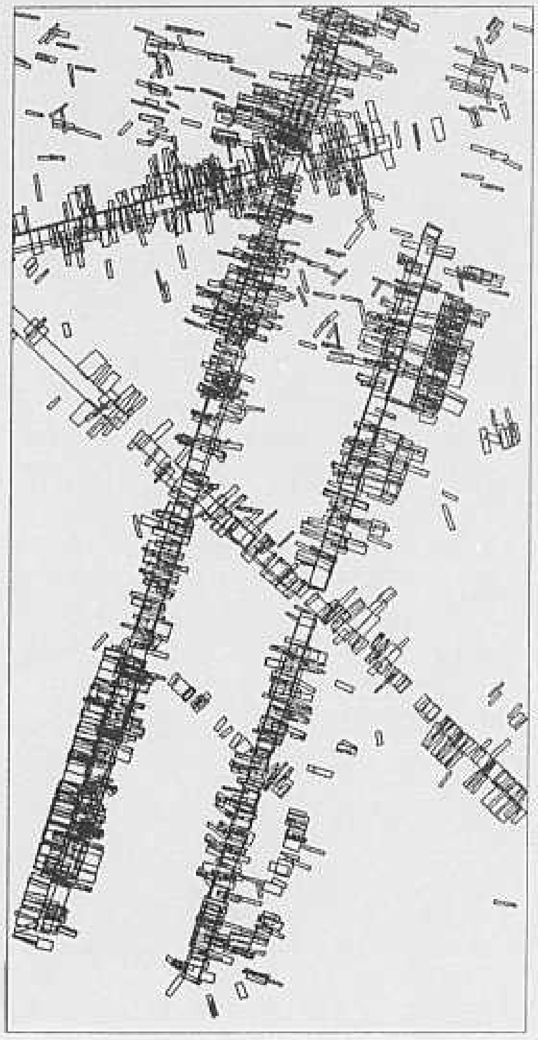


Figure 28: Initial set of Runway Fragments in LOGAN2

Figure 44 shows another portion (JFK2:2500 × 2000 resolution) of John F. Kennedy International Airport in New York. This airport scene poses numerous difficult problems. The changes in surface material due to repairs and expansion occurs randomly. Some of the expansion work consist of strips having different widths as the original runway, in addition of being of different material. The center strip, presumably an old runway, is as wide as other runways but has no discernible markings applied to it. It is also wider than the new runway on the left, which in turn, has non standard markings applied to it.

The line segments computed from the JFK2 image are shown in figure 45. The 44,340 apars computed from these segments are shown in figure 46. The estimates for runway directions are chosen to be the four peaks form the length weighted histogram of the orientation of the apars, shown in figure 47. The runway width estimates are obtained form the length weighted width histogram of the apars in the

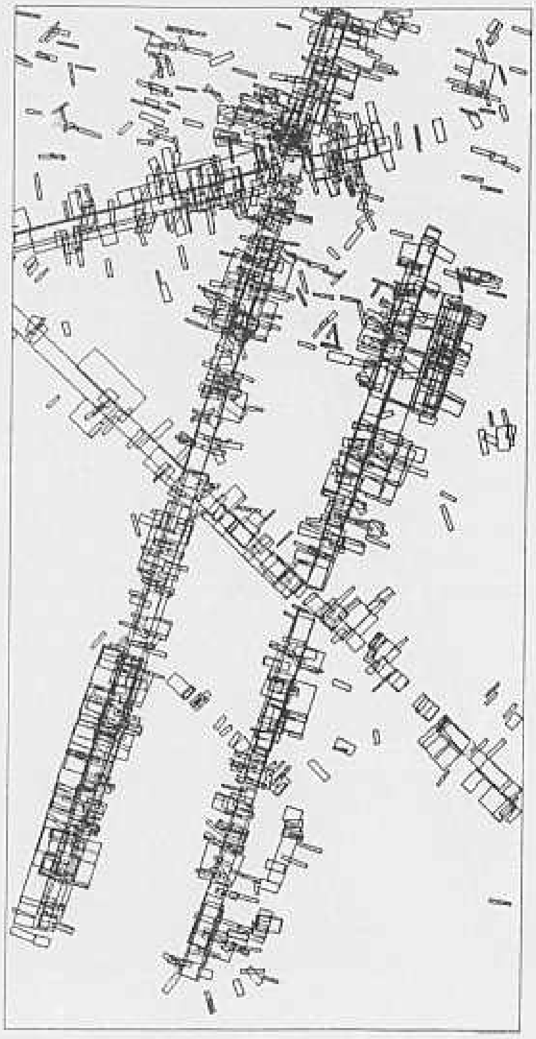


Figure 29: Apars in LOGAN2 after joining on continuity

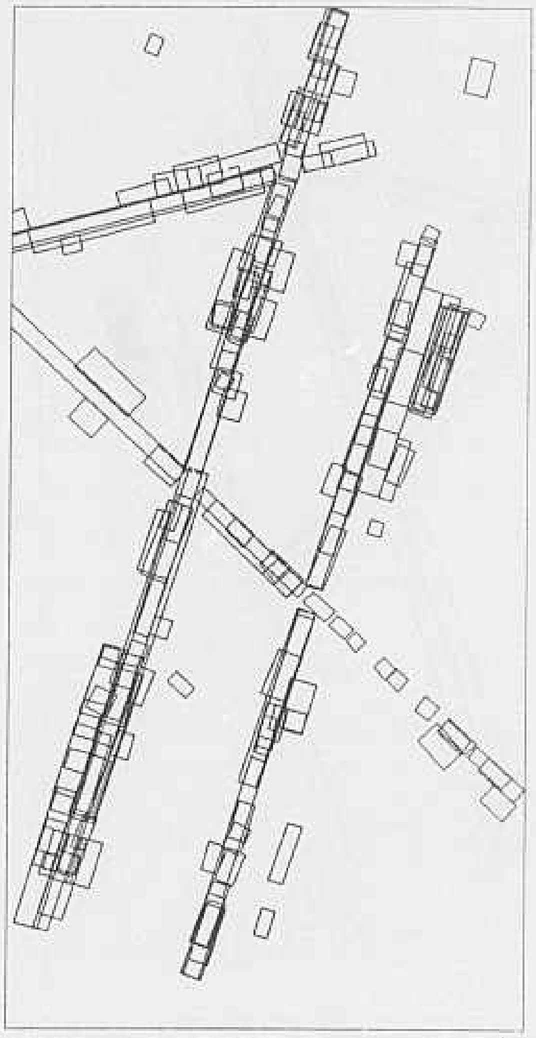


Figure 30: Apars in LOGAN2 after removal of short and contained apars

selected directions, as described previously. The reduced search space and apars representing the initial set of runway fragments is shown in figure 48.

Figure 49 shows runway apar fragments joined on boundary continuity. Figure 50 shows the aparar remaining after removal of the apars with an aspect ratio of less than one, and removal of properly contained apars. The apars are then joined on the basis of analysis of the gaps between fragments are shown in figure 51. The thresholded apars on aspect ratio give the runway hypothesis, shown in figure 52. Figure 53 shows the results of the verification process. In this example we show the runway hypotheses for which at least one type of marking appeared to be found.

5 Conclusion

The modeling of runways in major commercial airports is not as straightforward as it may seem at first. From the examples shown, we can infer that runways can be very complex objects to detect, analyze and describe in a useful manner for automated mapping and photointerpretation tasks.

We have described a technique, based on geometry and shape as the sources of knowledge suitable to form and test hypotheses representing instances of a known object shape, airport runways, using the line segments and anti-parallel pairs of line segments computed from the images.

Our work is part of a project to automatically map complex cultural areas such as a major commercial airport complexes. Our goal is to map all of the interesting objects in the scene and also to devise integrated descriptions that include the functional relationships of the objects in the scene. In this paper we concentrate on the mapping of runways (we are pursuing mapping of buildings in separate work [4,2]).

In work reported here, our verification step consists only of finding the various markings we expect. We have not yet combined the different criteria to give an overall confidence

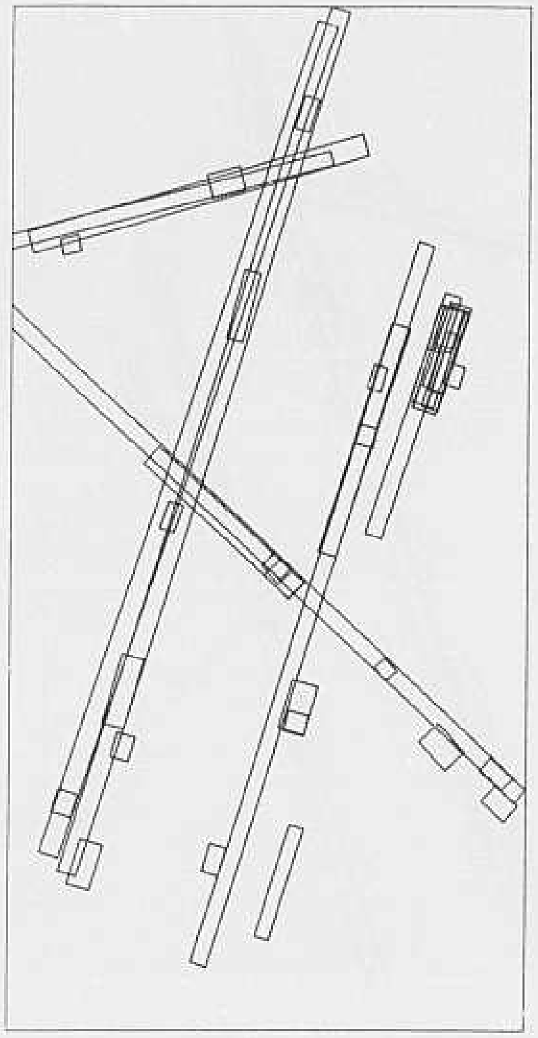


Figure 31: Apars in LOGAN2 after joining on gap analysis

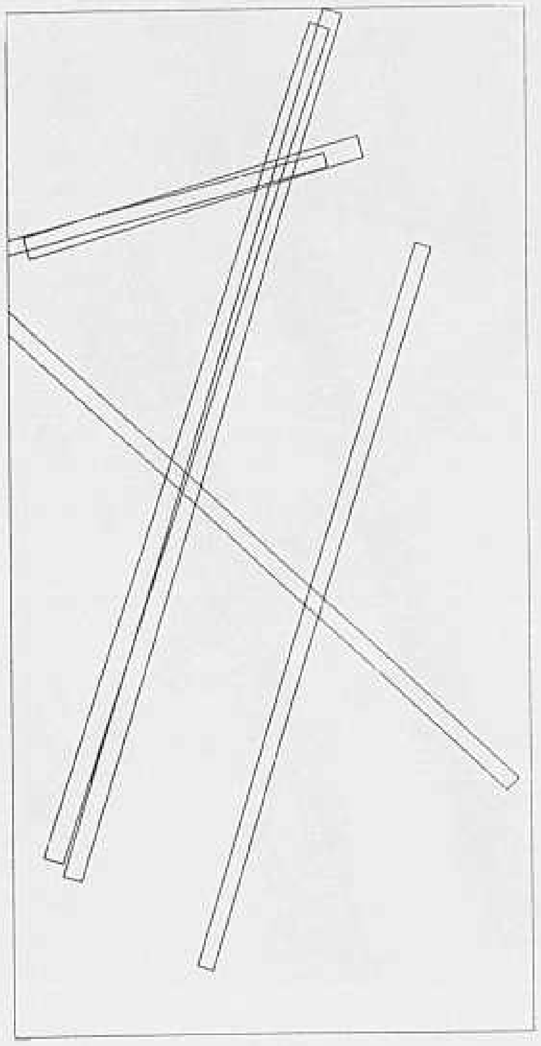


Figure 32: Final Runway Hypothesis for LOGAN2 image

value. This process should, ideally, take place in the context of the larger system that is also reasoning about other objects in the scene, such as the remainder of the transportation network, buildings and the mobile objects. Location of these objects will mutually affect the confidence levels of the descriptions of other objects. Thus, the system described here should be viewed as a module for the larger system to operate on.

We presented results on two very different airports to show the strength of the hypotheses formation process, based on linear features. Together with a sound search space reduction mechanism, and an object-specific feature verification technique, our method represents the state-of-the-art in runway detection. We have tested the technique on images of several major airports, varying in complexity between our two examples, with very encouraging results.

Our basic technique can be easily extended to use the intensity image if necessary, as well as for analysis of non-standard markings. We point out that our hypotheses formation/verification technique can be useful for similar tasks, such as road detection and in general, transportation network detection.

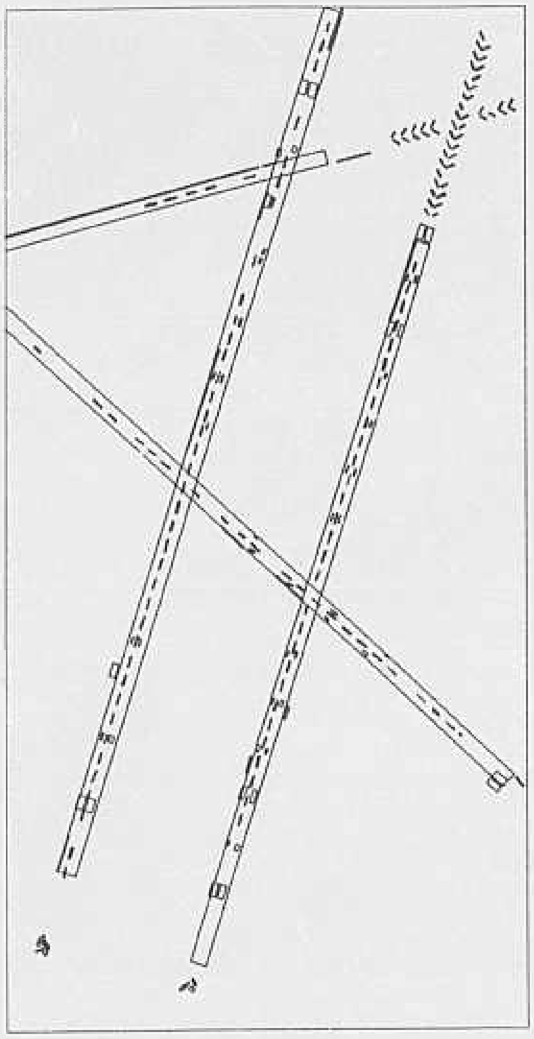


Figure 33: Selected Set of Markings for Verification of Hypothesis (LOGAN2)

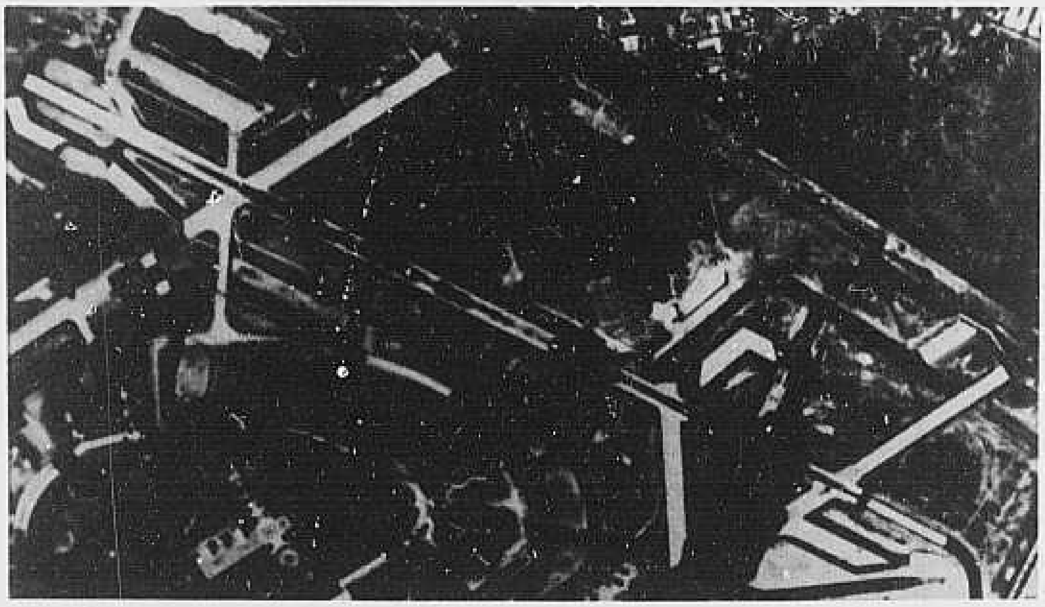


Figure 34: John F. Kennedy International Airport (JFK1) image

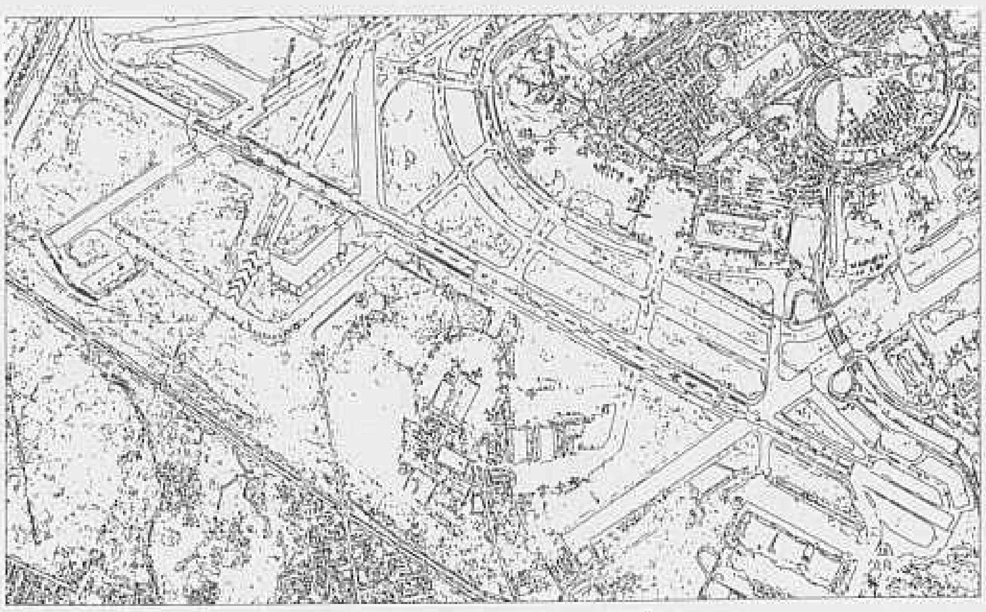


Figure 35: Segments computed from JFK1 image

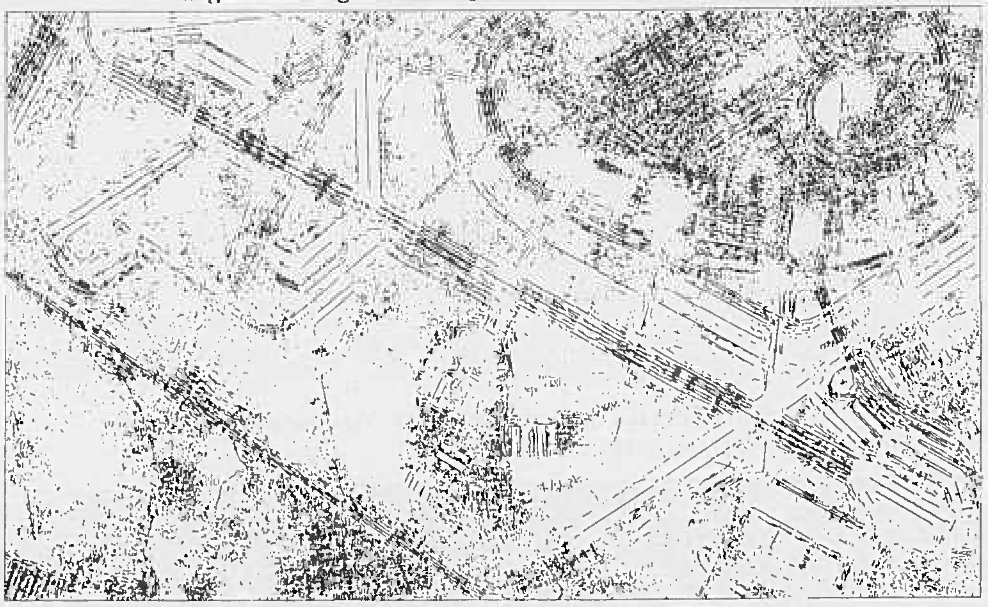


Figure 36: Anti-parallels from the segments in JFK1 image

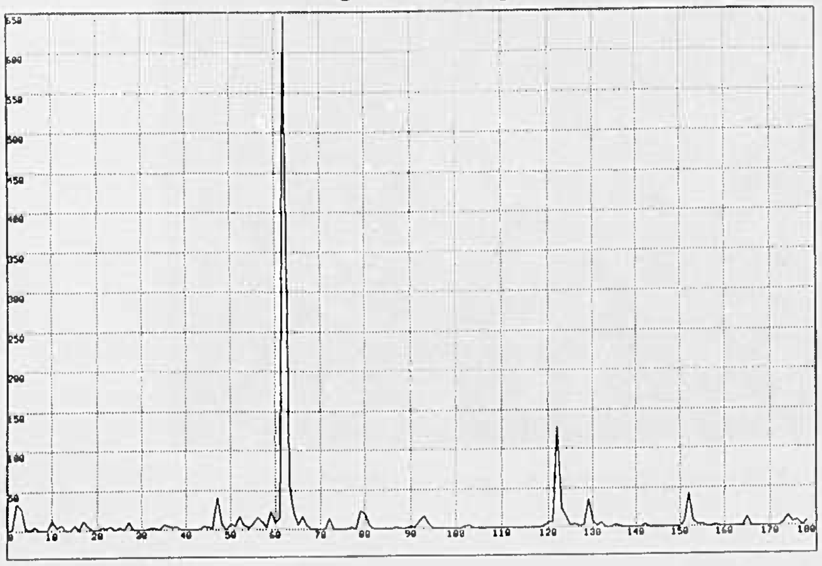


Figure 37: Length weighted apar orientation histogram for JFK1

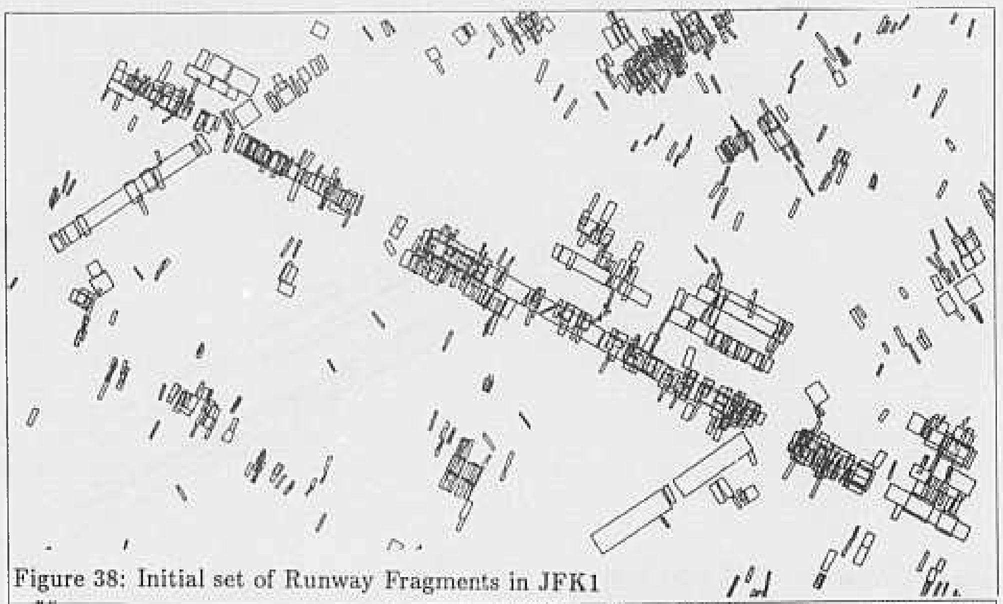


Figure 38: Initial set of Runway Fragments in JFK1

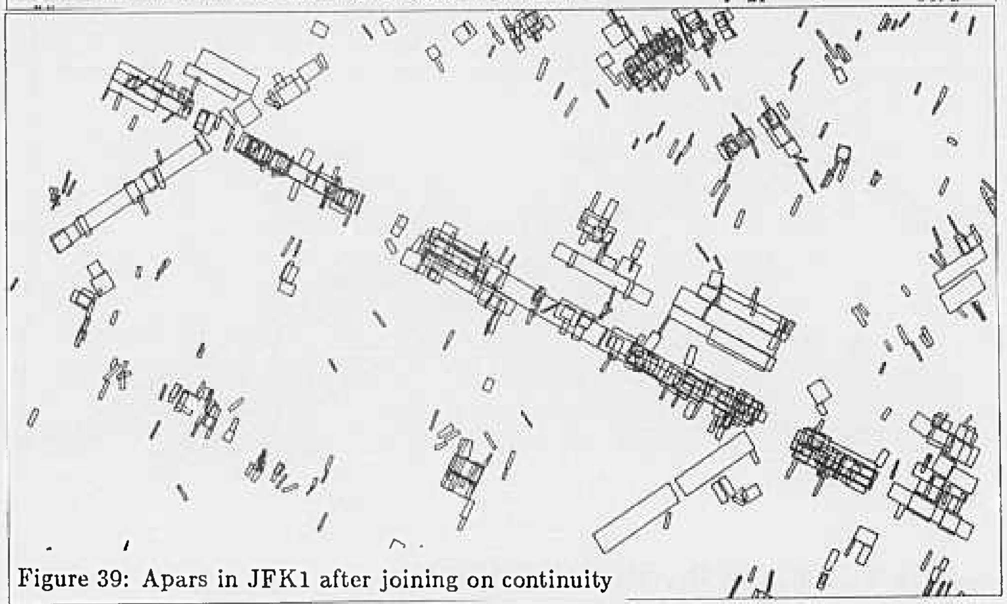


Figure 39: Apars in JFK1 after joining on continuity

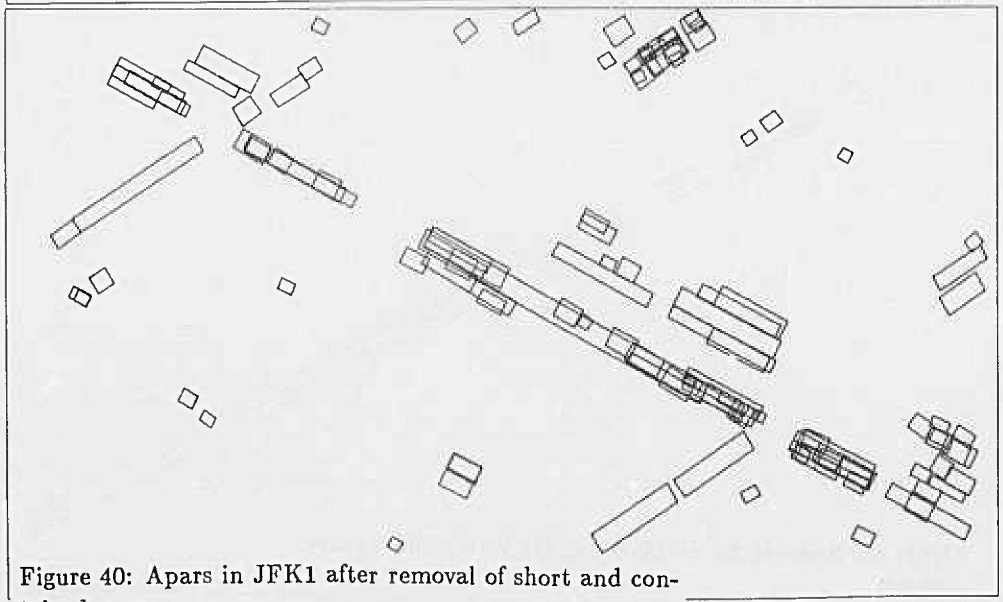


Figure 40: Apars in JFK1 after removal of short and contained apars

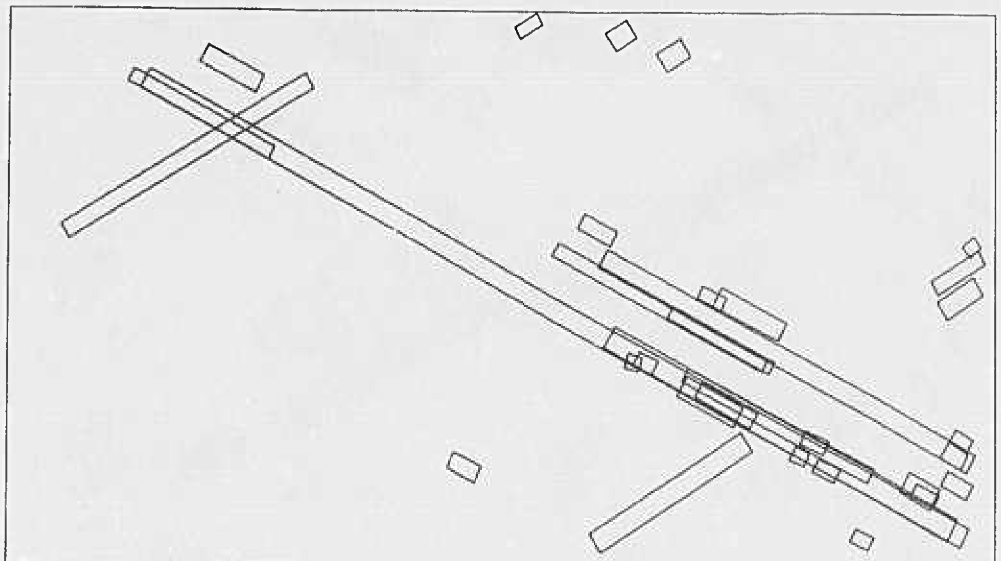


Figure 41: Apars in JFK1 after joining on gap analysis

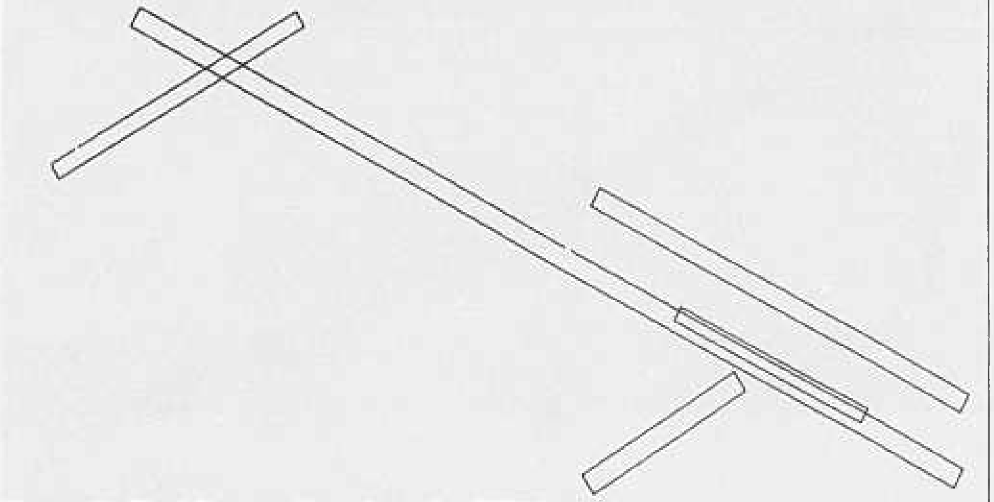


Figure 42: Final Runway Hypothesis for JFK1 image

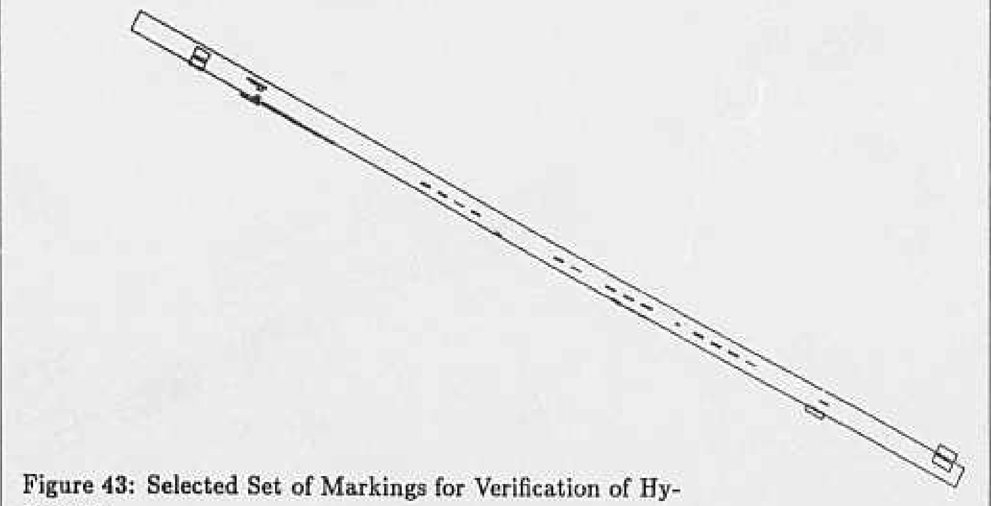


Figure 43: Selected Set of Markings for Verification of Hypothesis

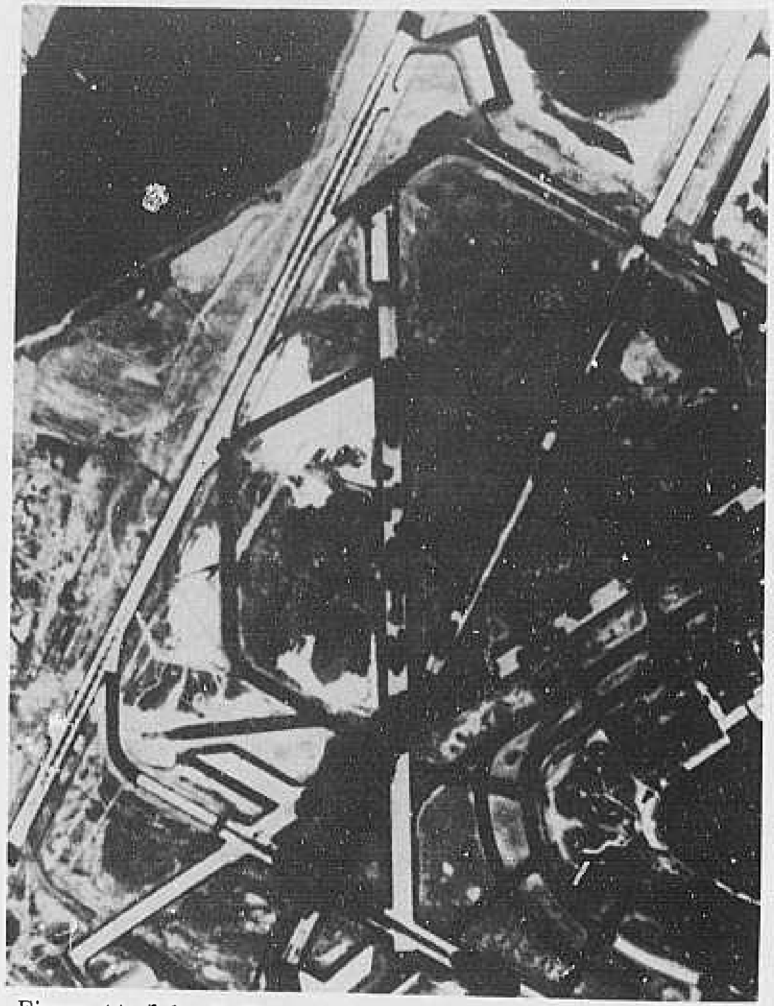


Figure 44: John F. Kennedy International Airport (JFK2) image

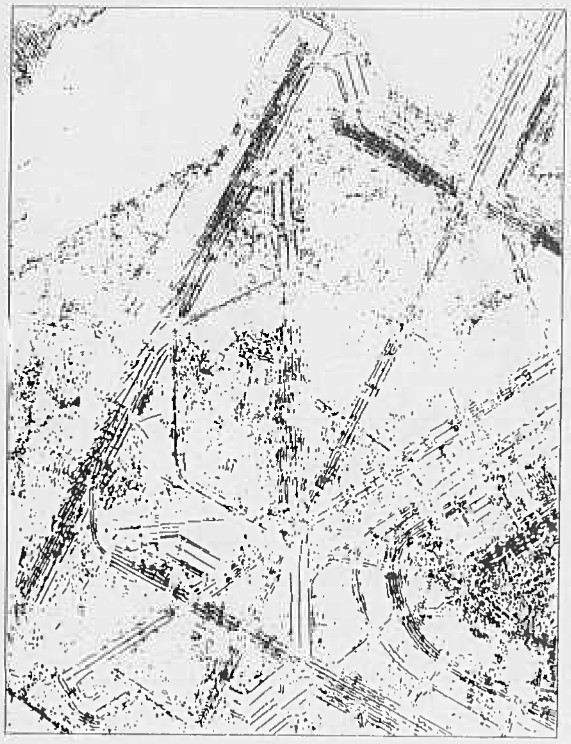


Figure 46: Anti-parallels from the segments in JFK2 image

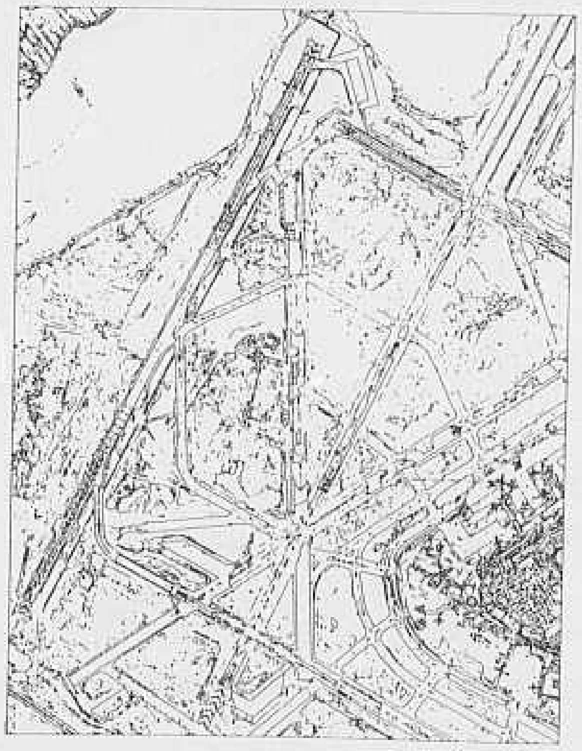


Figure 45: Segments computed from JFK2 image

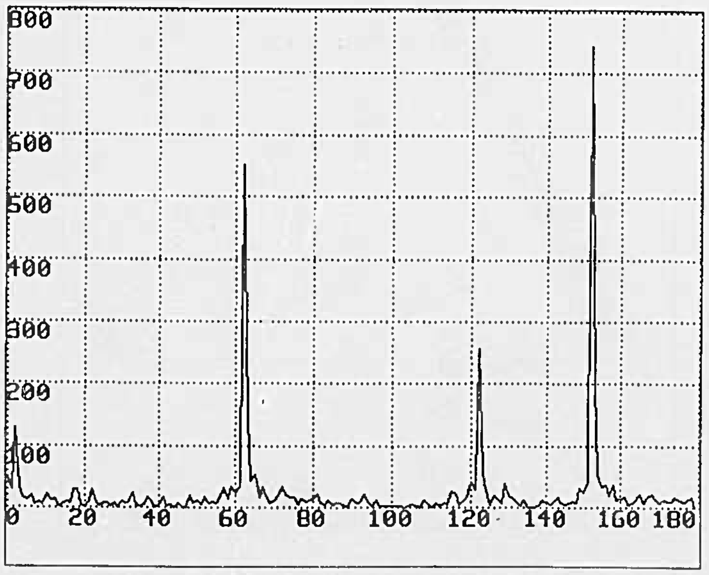


Figure 47: Length weighted apar orientation histogram for JFK2

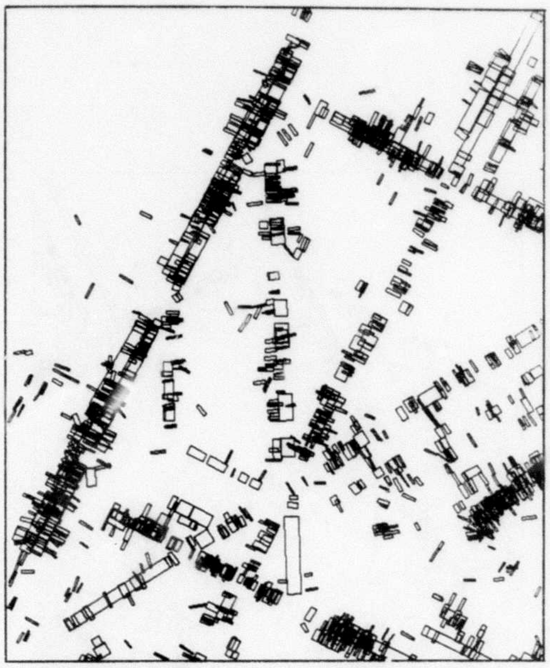


Figure 48: Initial set of Runway Fragments in JFK2

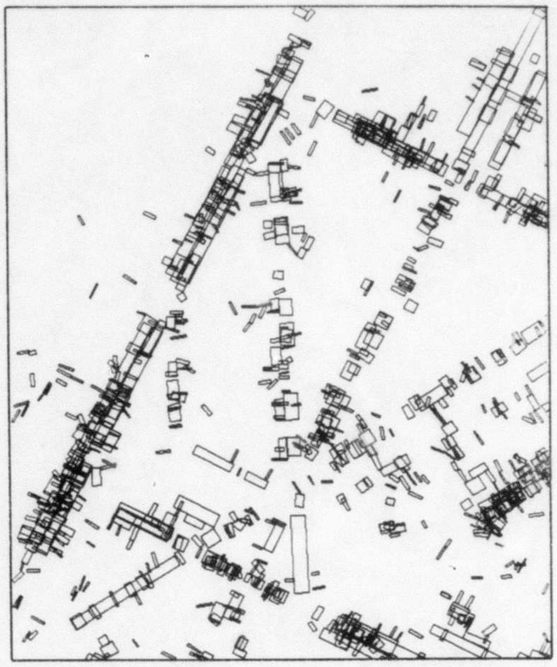


Figure 49: Apars in JFK2 after joining on continuity

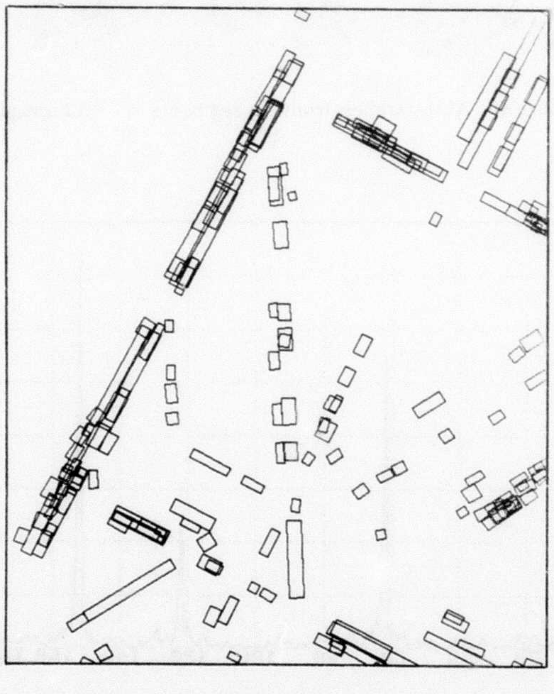


Figure 50: Apars in JFK2 after removal of short and contained apars

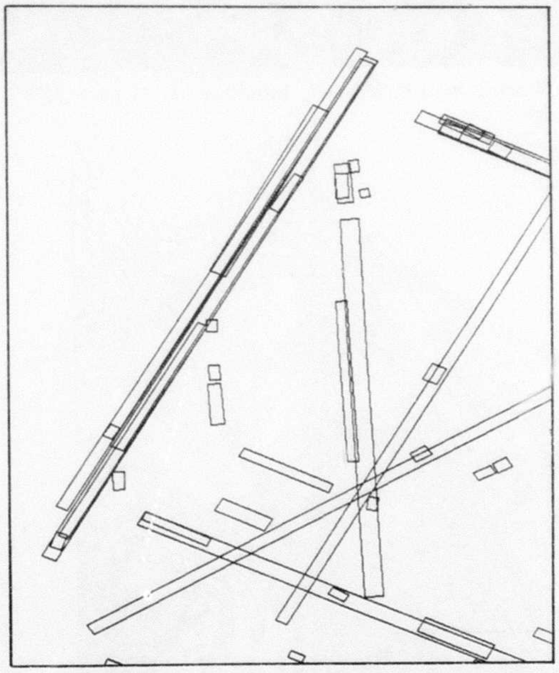


Figure 51: Apars in JFK2 after joining on gap analysis

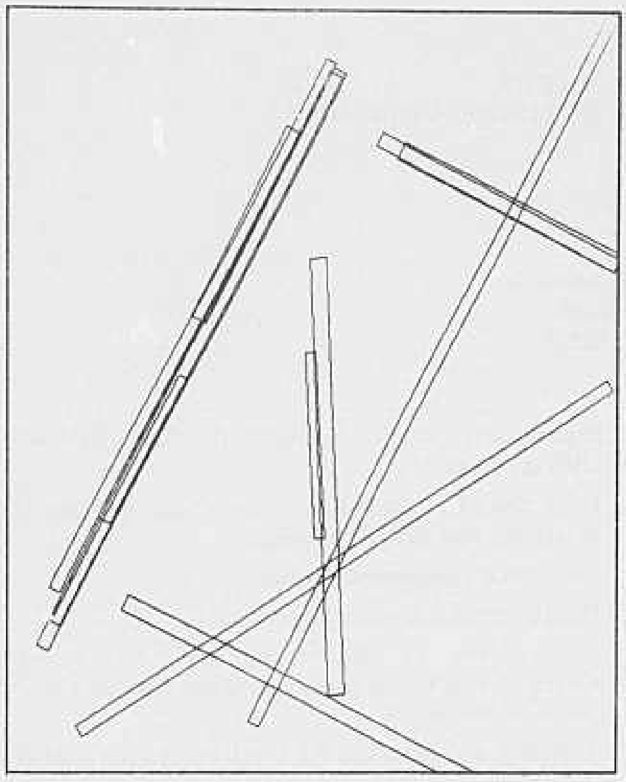


Figure 52: Final Runway Hypothesis for JFK2 image

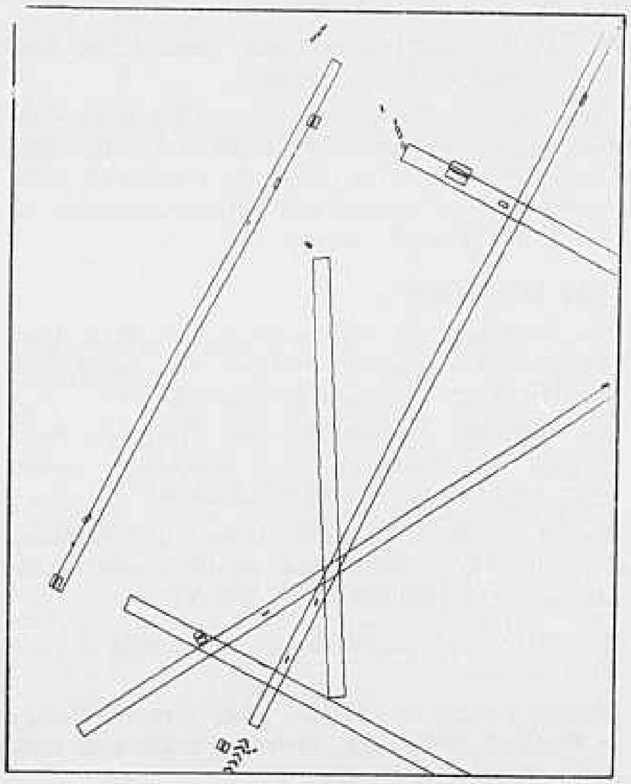


Figure 53: Selected Set of Markings for Verification of Hypothesis

References

[1] Nevatia, R. and Babu, R., "Linear Feature Extraction and Description", *Computer Vision, Graphics and Image Processing*, Vol. 13, 1980, pp. 257-269.

[2] Huertas A. and Nevatia, R., "Detection of Complex Buildings in Simple Scenes", *Technical Report IRIS No. 203*, Institute for Robotics and Intelligent Systems, University of Southern California, September 1986.

[3] Huertas A. and Nevatia, R., "Edge Detection in Aerial Images using $\nabla^2(x,y)$", *Technical Report ISCIPI No. 1010*, Image Processing Institute, University of Southern California, March 1981.

[4] Huertas, A. and Nevatia, R., "Detecting Buildings in Aerial Images". To appear in *Computer Vision, Graphics and Image Processing*

[5] Marr, D. and Hildreth, H.,"Theory of Edge Detection", *Proceedings of the Royal Society of London*, B207, 1980, pp. 187-217.

[6] Ashford, N. and Wright, P.H.,"Airport Engineering, 2nd Ed.", *Wiley and Sons*, 1984.

[7] "Marking Paved Areas on Airports", FAA Advisory Circular AC 150/5340-1E, November 4, 1980. *Wiley and Sons*, 1984.

A REPORT ON THE DARPA IMAGE UNDERSTANDING ARCHITECTURES WORKSHOP

Azriel Rosenfeld

Center for Automation Research
University of Maryland
College Park, MD 20742

ABSTRACT

This report summarizes the results of a workshop on Architectures for Imge Understanding (IU), held in McLean, VA on November 13-14, 1986. Benchmark results on a set of IU-related tasks were presented at the Workshop for eight architectures (the presenting organizations are given in parentheses): The Butterfly (University of Rochester), the Connection Machine (MIT), NON-VON (Columbia University), Message-passing concurrent computers (California Institute of Technology), the ENCORE MULTIMAX (University of Massachusetts), the Image Understanding Architecture (University of Massachessetts), Warp (Carnegie-Mellon University), and the Hierarchical Bus Architecture (Hughes AI Center). This report defines the tasks, briefly summarizes the benchmark results, and discusses criteria for the design of future IU benchmarks.

1. THE WORKSHOP

One of the goals of the DARPA Strategic Computing Program is to develop computer architectures for image understanding and computer vision. Many new architectures have been designed or built that can be used for IU tasks, but little is known about their relative capabilities. The DARPA Image Understanding Architectures Workshop made an initial attempt to gather information about the performance of some of these architectures on a set of IU-related computational tasks. This type of comparative study can lead to greater understanding of the types of architectures needed at various levels of the IU problem.

The DARPA program managers involved in the organization of the Workshop were Mark Pullen, Bob Simpson, and Stephen Squires.

1.1. The Benchmarks

A set of representative IU-related computational tasks was selected at a meeting of the principal investigators of the DARPA Image Understanding Program in Denver, CO on June 21, 1986. Benchmark problems based on these tasks were designed and distributed to the architecture designers. The detailed definitions of these benchmarks are given in Appendix A. They deal with the following tasks:

1) Edge detection (including convolution, zero-crossing detection, and border following)
2) Connected component labeling
3) Hough transform computation
4) Computation of the convex hull, the Voronoi diagram, and the minimal spanning tree of a set of points in the plane
5) Visibility computation for a set of opaque triangles in 3-space
6) Finding subgraphs of a given graph that are isomorphic to another given graph
7) Finding the minimum-cost path between two vertices of an edge-weighted group.

Tasks (1-3) deal with pixel arrays, tasks (4-5) with geometric (coordinate) data, and tasks (6-7) with relational structures. Thus the tasks span a range of problems representative of those that might be encountered at successive levels of the IU process.

1.2. The Architectures

The Workshop was help on November 13-14, 1986 at BDM International, Inc. in McLean, VA. Benchmark results were presented for eight architectures:

a) The Butterfly Parallel Processor (built by Bolt, Beranek and Newman, Inc.; benchmark results presented by the University of Rochester)
b) The Connection Machine (built by Thinking Machines Corp.; benchmark results presented by Massachusetts Institute of Technology)
c) The NON-VON supercomputer (Columbia University)
d) Message-passing concurrent computers (California Institute of Technology), including a 256-node cube and a 16,000-node Mosaic.
e) The ENCORE MULTIMAX parallel processing computer (built by Encore Computer Corp.; benchmark results presented by the University of Massachusetts)
f) The Image Understanding Architecture (University of Massachusetts)

g) The Warp Programmable Systolic Array Processor (Carnegie-Mellon University)

h) The Hierarchical Bus Architecture (Hughes AI Center)

Detailed descriptions of the benchmark results will not be given in this report. A table summarizing some of the results is given as Appendix B. However, caution should be exercised in using this table for comparison purposes. As discussed in Section 2.6, such comparisons are likely to be misleading because the results were not all obtained in consistent ways; in particular, some of them are based on simulations or estimates, while others are based on actual runs on real machines.

2. ISSUES

This section discusses the issues raised by and lessons learned from the benchmark exercise, and also makes some suggestions about future benchmarking experiments.

2.1. Research vs. applications

The immediate uses of new IU architectures will primarily be in research or exploratory development environments, not in practical applications. Thus it is not possible to predict what specific computational tasks these architectures will be called on to perform. Benchmarks for testing such architectures should therefore be based on representative computational tasks typical of those encountered at various stages of the vision process.

Performance standards for a research system are quite different from those for an application-oriented system. In a research environment, run times measured in minutes will often be acceptable, whereas real-time applications may require run times on the order of fractions of a second.

2.2. Goals vs. tasks vs. algorithms

Benchmarks should not be based on imprecisely defined goals, such as "detect the edges of the regions in the image". If such goals were allowed, benchmark evaluation would be difficult, since it would be necessary to take into account the quality of the results as well as the speed of the computation. Rather, the benchmarks should involve well-defined computational tasks, such as "apply the Sobel operator to the image". On the other hand, they should specify only the computation to be performed, but not the specific algorithm used to carry it out; for example, if the task is to find the lowest-cost path between two nodes in a network, the benchmark should not specify that Dijkstra's greedy algorithm be used. The preferred algorithm for a given task may depend strongly on the architecture, and it would not be desirable to constrain the algorithm choice. At the same time, it would be very desirable to obtain insights into the types of algorithms and types of computational tasks that a given architecture supports.

2.3. Benchmark sizing

The computational tasks normally encountered in IU systems do not span a large range of scales. Images are usually of standard sizes, e.g. 512×512 one-byte pixels. This constrains the sizes and complexities of the geometrical entities (borders, connected components, feature point sets) derived from the images, which in turn constrains the sizes of the graphs representing relations among these entities. It may also be acceptable to limit the precision of the computations performed on these structures, e.g. to use integer rather than real arithmetic. The benchmarks prepared for the workshop were not all consistent in these respects.

It is useful to distinguish between worst-case and typical-case performances on the benchmarks. Future benchmark experiments should make use of specific input data sets (images, graphs, etc.), to make it easier to compare typical-case results. The data sets should span the range of expected complexities, so that performance degradation as a function of complexity can be assessed.

2.4. Benchmark selection

The benchmark tasks used in the workshop represented only a small fraction of the operations that might be used in an IU system. Hundreds of different techniques can be used for image preprocessing, feature extraction, segmentation, and property measurement. Techniques for recovering three-dimensional scene information from an image or image sequence ("shape from x" and "structure from motion" techniques) were not represented in the benchmarks at all. Only two simple methods of representing geometric entities extracted from images (binary images and lists of point coordinates) were used; other representations, such as run length codes, border codes, medial axis transformations, or quadtrees, were ignored, as were three-dimensional object and surface representations.

The highest level of abstraction used in the benchmarks was that of graph structures (presumably derived from an image). Reasoning and inference played no role. It would be desirable to define benchmarks that represented the reasoning level of the IU process.

2.5. Benchmark organization

Each of the workshop benchmarks was defined as a self-contained task; no attempt was made to structure the benchmarks into a coherent sequence of operations involving (at least) images, geometric data, and relational structures. Future benchmark experiments should include such sequences of tasks. This would raise significant issues of data remapping, and would allow the input/output requirements of the tasks, as well as their computational costs, to be evaluated more realistically.

Another consequence of the use of self-contained benchmarks is that the tasks were highly homogeneous. This usually made it possible to efficiently partition the data among the processors, with little or no contention for resources and little need for communication among processes. A more challenging class of benchmarks would involve multiple processes operating on different parts of the image, or on different sets of image-derived data. This would greatly inerease the level of interprocess eommunication and contention, and would lead to significant problems of top-down eontrol. The difficulties arising at this level may be the real limiting factors in the performance of vision systems.

2.6. Benchmark evaluation

As mentioned at the end of Section 1, it is difficult to draw definitive conclusions from a eomparison of the benchmark results. Some of the participants in the benchmark exercise were using real machines, while others were using simulations or even estimates. Some of them counted only aetual running time on the task, while others included various overhead items such as input/output time, compilation time, downloading time, etc. Some gave worst-ease results while others gave "average" results, which sometimes depended on the speeifie input data that were used. In some cases the results could have been improved by a different choice of algorithms; many of the benehmarks were implemented under tight time constraints.

Another serious comparability issue relates to computation time vs. programming time. If a system is to be used in a researeh environment (Section 2.1), ease of programming is at least as important as computational speed. Future benchmarks should obtain information about the programming effort involved, as well as the computational eost.

3. CONCLUDING REMARKS

It was generally agreed by the participants that the benehmarking exercise as a sueeess, in spite of all its limitations. It forced the architecture designers to address a common set of tasks and to produce concrete results, and it thus provided some basis, however imperfect, for assessing the strengths and weaknesses of the various systems. Future benchmarking experiments, designed in aecordance with the guidelines suggested in Seetion 2, are planned.

APPENDIX A. DEFINITIONS OF THE BENCHMARKS

This appendix eontains the benchmark descriptions that were distributed to the Workshop participants in August 1986.

(1) Edge detection
In this task, assume that the input is an 8-bit digital image of size 512×512 pixels.

a) Convolve the image with an 11×11 sampled "Laplaeian" operator [1]. (Results within 5 pixels of the image border can be ignored.)

b) Deteet zero-erossings of the output of the operation, i.e. pixels at whieh the output is positive but which have neighbors where the output is negative.

e) Sueh pixels lie on the borders of regions where the Laplacian is positive. Output sequenees of the coordinates of these pixels that lie along the borders (On border following see [2], Section 11.2.2.)

(2) Connected component labeling
Here the input is a 1-bit digital image of size 512×512 pixels. The output is a 512×512 array of nonnegative integers in which

a) pixels that were 0's in the input image have value 0

b) pixels that were 1's in the input image have positive values; two sueh pixels have the same value if and only if they belong to the same connected component of 1's in the input image.

On eonnected eomponent labeling see [2], Section 11.3.1.)

(3) Hough transform
The input is a 1-bit digital image of size 512×512. Assume that the origin (0,0) is at the lower left-hand eorner of the image, with the x-axis along the bottom row. The output is a 180×512 array of nonnegative integers constructed as follows: For each pixel (x,y) having value 1 in the input image, and each i, $0 \leq i < 180$, add 1 to the output image in position (i,j), where j is the perpendicular distance (rounded to the nearest integer) from (0,0) to the line through (x,y) making angle i degrees with the x-axis (measured eounterclockwise). (This output is a type of Hough transform; if the input image has many eollinear 1's, they will give rise to a high-valued peak in the output image. On Hough transforms see [2], Seetion 10.3.3.)

(4) Geometrical constructions
The input is a set S of 1000 real coordinate pairs, defining a set of 1000 points in the plane, selected at random, with eaeh eoordi- te in the range [0,1000]. Several outputs are requir

a) An ordered list of the pairs that lie on the boundary of the convex hull of S, in sequence around the boundary. (On eonvex hulls see [3], Chapters 3–4.)

b) The Voronoi diagram of S, defined by the set of coordinates of its vertiees, the set of pairs of vertices that are joined by edges, and the set of rays emanating from vertiecs and not terminating at another vertex. (On Voronoi diagrams see [3], Section 5.5.)

c) The minimal spanning tree of S, defined by the set of pairs of points of S that are joined by edges of the tree. (On minimal spanning trees see [3], Section 6.1.)

(5) Visibility
The input is a set of 1000 triples of triples of real coordinates $((r,s,t),(u,v,w),(x,y,z))$, defining 1000 opaque triangles in three-dimensional space, selected at random with eaeh eoordinate in the range [0,1000]. The output is a list of vertices of the triangles that are visible from (0,0,0).

(6) Graph matching
The input is a graph G having 100 vertices, each joined by an edge to 10 other vertices selected at random, and another graph H having 30 vertices, each joined by an edge to 3 other vertices selected at random. The output is a list of the occurrences of (an isomorphic image of) H as a subgraph of G. As a variation on this task, suppose the vertices (and edges) of G and H have real-valued labels in some bounded range; then the output is that occurrence (if any) of H as a subgraph of G for which the sum of the absolute differences between corresponding pairs of labels is a minimum.

(7) Minimum-cost path
The input is a graph G having 1000 vertices, each joined by an edge to 100 other vertices selected at random, and where each edge has a nonnegative real-valued weight in some bounded range. Given two vertices P,Q of G, the problem is to find a path from P to Q along which the sum of the weights is minimum. (Dynamic programming may be used, if desired.)

REFERENCES

(1) R.M. Haralick, Digital step edges from zero crossings of second directional derivatives, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **6**, 1984, 58–68.

(2) A. Rosenfeld and A.C. Kak, *Digital Picture Processing* (second edition), Academic Press, New York, 1982.

(3) F.P. Preparata and M.I. Shamos, *Computational Geometry—An Introduction*, Springer, New York, 1985.

APPENDIX B. SUMMARY OF BENCHMARK TIMINGS

The timing data given below were taken from the benchmark reports that were prepared by the participants and distributed at the Workshop. They should not be used uncritically for comparison purposes, because they were not all obtained in consistent ways, as indicated in Section 2.6. Timings are not given in cases where the test that was performed did not conform to the benchmark definition given in Appendix A. In particular, no timings are given for the subgraph isomorphism benchmark, the definition of which involved an impractically costly computation. If a participant gave several timings for a given task, only the shortest one is given here.

The column numbers in the table below refer to the following benchmark tasks:

1a–b)	Edge detection (convolution and zero crossing detection)
1c)	Border following
2)	Connected component labeling
3)	Hough transform
4a)	Convex hull
4b)	Voronoi diagram
4c)	Minimal spanning tree
5)	Visibility
7)	Minimum-cost path

The timings are all given in seconds; they are rounded to (at most) three significant digits.

Architecture	Task/time (seconds) (1a–b)	(1c)	(2)	(3)	(4a)	(4b)	(4c)	(5)	(7)
Butterfly									
16 nodes	18.3	–	–	45.2	0.19	–	–	67	–
100+ nodes	2.9	8.2	7.2	7.4	–	–	–	4.15	–
Connection Machine	0.003	0.1	0.4	0.7	0.2	2	2.2	1	0.05
NON-VON	0.002	–	1	0.4	0.04	–	0.04	0.1	0.04
Cube (256 nodes)	0.1	–	0.014	1.8	0.0024	–	–	–	0.01
Mosaic (16K nodes)	0.0025	0.001	0.006	0.01	0.0036	–	–	–	0.001
ENCORE MULTIMAX (20 nodes)	46.0	6.9	22.7	244	1.8	30.0	8.7	91.4	0.18
Image Understanding Architecture	0.00002	0.0002	0.000005	0.027	0.007	0.050	0.011	0.02	0.0007
Warp	0.016	0.69	0.075	0.60	0.003	0.011	0.043	0.040	0.025
Hierarchical Bus Architecture									
16 nodes	3.2	0.1	0.17	0.27	0.55	–	–	–	–
100+ nodes	0.6	–	0.37	0.03	0.94	–	–	–	–

IMAGE PROCESSING TO GEOMETRIC REASONING: MILITARY IMAGE ANALYSIS AT GE FESD

M. S. Horwedel
General Electric Federal Electronic Systems Division
P. O. Box 8555, Philadelphia, PA 19101

Abstract

This paper presents an overview of computerized image analysis efforts within General Electric's Federal Electronic Systems Division (FESD; formerly Space Systems Division.) It begins by outlining image analysis work which predated the military work (primarily in earth resources analysis), shows how this led to the early work in support of military applications, and describes the current directions of research within FESD.

1 Introduction

This paper presents an overview of efforts within the Federal Electronic Systems Division (Formerly Space Systems Division) which have involved the application of image analysis methods in support of military operations. The first section outlines some of the work which predates the current work in military applications. The second section discusses the Radar Interpreter's Aid (RIA) project and the IR&D work which preceded awarding of the RIA contract. The final section describes the current work in model directed image understanding and in building dedicated hardware to accelerate image analysis tasks.

2 Precursors

The early work in image analysis within Space Division was in support of the earth resources community. Our Lanham, Maryland facility was a key supporter of NASA's Goddard facility in developing processing methods used with the LANDSAT system. In addition, the Image 100 system, which was designed to permit fast interactive analysis of multispectral images, was designed and built at General Electric's Daytona Beach facility. The Image 100 system won an Industrial Research Magazine award in 1974 as one of the 100 most significant new technical products (reference [1]).

In using the Image 100, one first loads the image to be analyzed. This can be done by loading a computer tape containing a digitized image, or by using the input scanner unit to digitize a hardcopy image. The image is then preprocessed to correct for non-uniform illumination. Following the shading correction, the operator can specify one of four channel ratioing options. The four-channel data can then be transformed using Hadamard, eigenvector, or manual rotation. The manual rotation is performed using a joystick, while observing the results interactively. Three different options are then available for contrast enhancement. The user then interactively indicates the training sets by identifying ground truth correspondences with image areas. The actual classification can then be performed on up to 16 pixel classes (with appropriate hardware options) using linear discriminants or maximum likelihood Gaussian classification. The Image 100 is still widely used in earth resources work, including the work done at our Lanham, Maryland site.

3 Early Military Work

This section reviews the early work in image analysis within FESD, starting with theoretical work on statistical pattern matching for radar data and culminating in the RIA project, which resulted in the delivery of a program which combined an expert system for Synthetic Aperature Radar (SAR) with a number of image processing modules which the expert system could invoke.

3.1 Early Radar Work

Because the work in military applications of image analysis within Space Systems Division grew out of the LANDSAT work, it was strongly influenced by the processing paradigms which were so successful in the LANDSAT work. Thus, the early investigations focused heavily on texture measures. Since the use of texture measures provided ways to form multivariable feature vectors at the pixel level, the multispectral classification methods used in earth resources work could be applied to the imagery our military customers were interested in analyzing, which was generally not multispectral. This approach was initiated by the work of Dr. L. Alexander (reference [2]). This work focused on the extraction of information using both raw radar backscatter data and visible images, using multi-feature classification on a pixel basis. The feature vectors are formed by using various texture measures. Although this work did not make use of military imagery, it was motivated by the needs of our military clients, and much of the work in support of the military within FESD made heavy use of his results. The investigation covered 104 different image texture measures, as well as 390 different "roughness" measures which were applied to radar backscatter data prior to image formation. The taxonomy of the visual measures is:

- Co-occurrence measures
- Fourier Transform Measures
- Differences of Averages (edgeness)
- Gray Level Run Length Characterization
- Autocorrelation Function Features
- Mitchell's Max-Min Measure

The "microwave roughness measures" are a parameterizations of various cuts through th curve of microwave cross-section/unit are σ, as a function of frequency, angle of inc dence, and polarizations of the transmittin and receiving antennas. The ability of each these measures to discriminate among the Re idential, Agricultural, Forest, and Water pix classes was investigated.

Optimal performance was obtained by usi a combination of raw backscatter and visib image data. Specifically,

- If limited to a single feature, it is best choose the best "roughness" feature,
- If limited to a pair of features, a go "roughness" feature and a good "visu texture" feature should be employed, a
- If using three features, two "roughne features and one "visual texture" featu should be used.

This work led to much investigation of the use of the texture measures used in this report on images of military interest, both radar and optical. The radar work focused on the visual texture measures rather than the more effective raw backscatter "roughness measures" because of the difficulty of obtaining raw backscatter data of militarily interesting targets. The radar work culminated in the contract described in the next subsection.

3.2 The Radar Interpreters' Aid

The Radar Interpreters' Aid (RIA) was produced under a study contract as a "proof of concept" for the idea of combining an expert system with a library of signal processing algorithms. It produced a system capable of recognizing objects of military significance in a Synthetic Aperature Radar (SAR) image. Figure 1 shows the architecture of the RIA system. The expert system inference engine for the RIA project was DELTA, developed by General Electric's Corporate Research and Development center in Schenectady, New York for a system which diagnosed problems with diesel-electric locomotives. The rule base for the RIA system was written by FESD personnel, and includes rules in these general classifications:

- Modeling rules, which predict the appearance of a SAR image as a function of imaging geometry,
- Matching rules, which make the decision as to whether an area of the image is close enough in appearance to a modeling rule output to justify being classified, and
- Control rules, which invoke image processing routines as needed in support of the other rule types and provide interface to the user and peripheral devices.

The image processing library consists of image processing routines developed at FESD under IR&D funding. It built upon our library of texture routines, and added routines for

- Edge Detection, non-maximum suppression, and linking,
- Blob detection and grouping, and
- High interest subimage identification.

The RIA user session begins with the introduction of the fact that there is an image to analyze. The control rules then take over, invoking the image processing rules for high interest subimage indentification. These in turn invoke image processing routines which classify concrete areas using texture analysis, and decide which concrete areas are runways. Once the runways are localized, high-interest subimages are identified using heuristics about where airplanes are likely to be parked. These high-interest subimages are extracted from the input image, and are subjected to further processing to actually search for airplanes.

The modeling rules cover two types of "low-level primitives," which are dependent upon the imaging geometry:

- A collection of linear features, or
- A collection of blobs.

Figure 2 is a reconstruction showing what an optical image, a linear SAR image, and a blobby SAR image look like. Processing to detect airplane images with linear features is based upon detected edges in the image. Once edges are detected and linked into edge groups by the image processing modules, the rule base applies hueristics to detect the airplane. The sequence is: searching for a fuselage, then for wings and tail based upon the fuselage orientation, and finally for smaller features such as engines.

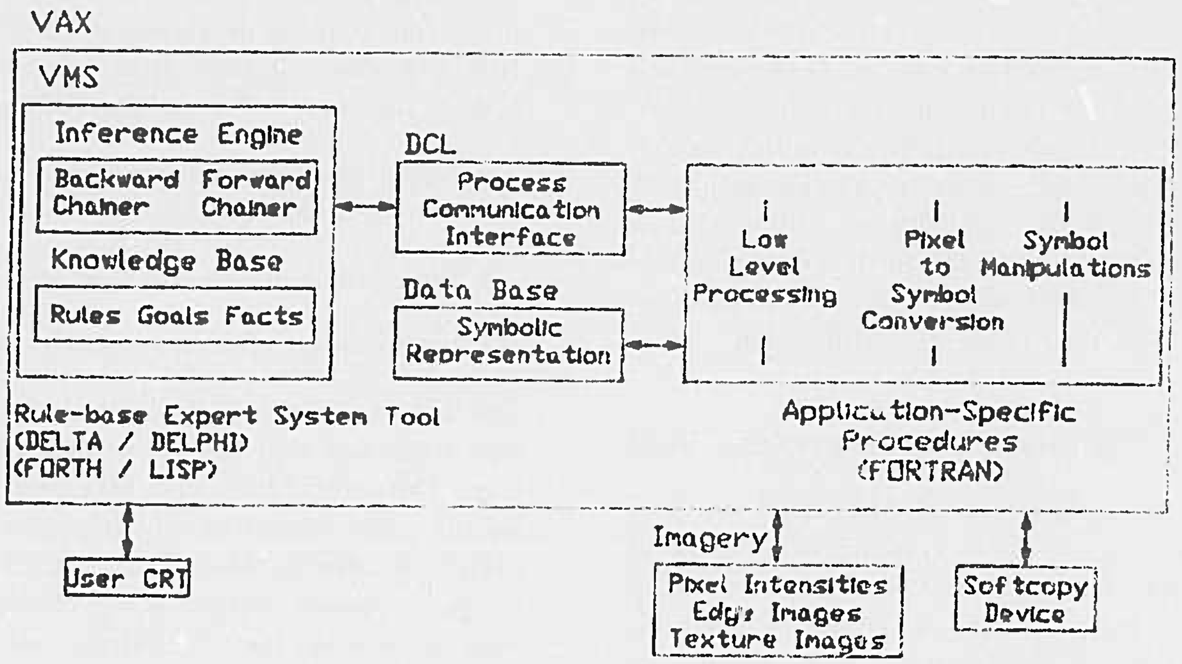


Figure 1: RIA Architecture

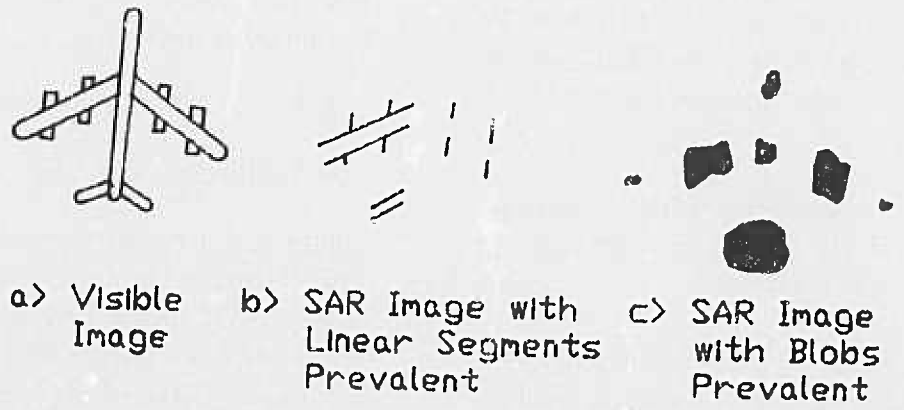

a) Visible Image
b) SAR Image with Linear Segments Prevalent
c) SAR Image with Blobs Prevalent

Figure 2: Sketches of Characteristic Optical and SAR Images

Processing to detect blobby airplane images is similar, except that the primitive features used are blob groups instead of edge groups. The image processing routines process blobs using a non-linear speckle-reducing filter, then use heuristics to try to group them into quasi-linear groupings. From the grouping step on, processing for blobby images is similar to that for linear images, except that tolerances for angle matches is increased, due to the fact that blob groups can not be characterized by angles as accurately as groups of detected edges.

The RIA project was successful, finding 12 of 13 airplanes with no false alarms in its operational test for the customer within 40 minutes on a 512^2 image. It demonstrated robustness in its ability to handle images with in-plane arbitrary rotations of the image airplanes. It was also successful in what it taught us in the way of suggesting improvements for further work, which are paraphrased from reference [3]:

- Use a system environment in which all of the software can be developed in the same language: RIA had an inference engine written in FORTH and signal processing languages written in FORTRAN, which led to implementation difficulties in integrating the modules.

- Consider the use of special-purpose hardware to accelerate either the image processing or the image understanding modules.

- Consider the use of a geometric model-based paradigm rather than a rule-based "expert system" as a means of packaging the "intelligence" for the modeling and matching functions.

The following section tells of our current efforts in military image analysis, and reflects the agenda which was enumerated above.

4 Current Work

This section describes work currently in progress on military applications of image analysis within GE FESD:

- Our tie into the research which is being done at the Corporate Research and Development Center in Schenectady, and

- The development of a hardware accelerator for pixel and neighborhood operations at the Valley Forge location.

4.1 Model Directed IU

Since the completion of the RIA project, FESD has been collaborating closely with Dr. J. Mundy of General Electric Corporate Research and Development Center. FESD is leveraging the work done by Dr. Mundy under DARPA contract and under internal IR&D funding into FESD's image understanding applications. The theoretical aspects of this work will be presented in other papers at this conference, so this discussion will center on the benefits we hope to realize including:

- A development environment which not only permits us to use a single language (LISP) for all software development, but which makes interactive experimentation with images and three-dimensional polyhedral models a reality,

- A geometric model-directed matching system which accomplishes what we set out to accomplish on the RIA project more robustly and in a more natural manner (references [4,5]), and

- Methods to automate the creation of the models which are needed for the model matcher (reference [6]).

Thus, our collaboration has addressed two of three areas identified for improvement in the

RIA project, as well as a real problem not previously considered. We are addressing the third problem area in the Feature Extraction Processor (FExP: pronounced as "fexpy") project.

4.2 The FExP Project

The FExP project was established to provide a low-cost means of improving the speed of at least some of our image analysis algorithms by two orders of magnitude at a cost which is less than that of commercial array processors. We chose to attack edge detection and related algorithms, because we had some relatively mature algorithms to work with, and did not want to commit to a hardware development with "experimental" algorithms. Another motivation was that the matching work at Corporate Research and Development was using edge lists as inputs. These algorithms take considerable time to compute, which was limiting the ability to experiment on a large enough set of images to statistically characterize the performance of our IU system. The solution selected was a hardware implementation using a connectionist architecture, consisting of a SIMD rectangular mesh of bit serial processors. Specifically, the prototype FExP is being built using Geometric Arithmetic Parallel Processor (GAPP) chips (reference [8]) to provide the processing power. In addition to meeting the stated goals of the project using this architecture, the following benefits influenced the decision:

- This architecture appeared applicable to a wider range of image analysis algorithms than other alternatives meeting the stated objectives for edge detection,
- The algorithms did not have to be "hard-wired," making it possible to reprogram the same machine for other tasks,
- This is an all-digital approach, which removes concerns about component tolerances and drifting parameters,
- The bit-serial architecture allows precision for speed tradeoffs to any degree desired, and
- Project members would get experience in one of the programming paradigms necessary to utilize the XΩ Machine being developed at GE Corporate Research and Development Center under DARPA contract (reference [9]) prior to the availability of the XΩ Machine.

The connection with the XΩ Machine project is proving very beneficial, providing leverage both ways.

The completed design consists of six VME boards, three off-the shelf, and three custom designed. We are currently fabricating the first of the custom boards, and have scheduled project completion by April of this year. Included in this projection is firmware for the following algorithms:

- The Canny Edge Detection Algorithm,
- The Fu Topology-Preserving Thinning Algorithm,
- A Region Growing Algorithm for line labeling, and
- An algorithm to break edge segments into linear features.

The completed FExP will be able to serve several hosts via a TCP/IP LAN connection. It is anticipated that three GE sites will be using FExP's to accelerate their image analysis research by the end of 1987.

References

[1] *Image 100 Interactive Multispectral Image Analysis System: System Description*, General Electric, Daytona Beach, FL, March 1975.

[2] L. Alexander. *Computer Information Extraction from Radar Images*, Technical Information Series No. 81SDS-034, General Electric, Valley Forge, PA, December 1981.

[3] P. T. Farnum. "AI Applications in Synthetic Aperature Radar Image Understanding," *AFCEA Intelligence Symposium on Imagery*, October 8-9, 1985.

[4] D. W. Thompson and J. L. Mundy. "Three Dimensional Model Matching From an Unconstrained Viewpoint", to be presented at *IEEE Conference on Robotics and Automation*, April 1987.

[5] D. W. Thompson and J. L. Mundy. "Model Directed Object Recognition on the Connection Machine," *DARPA Image Understanding Workshop*, February 1987.

[6] D. Cyrluk et. al. "The Formation of Partial Three-Dimensional Models from Two-Dimensional Projections—An Application of Algebraic Reasoning," *DARPA Image Understanding Workshop*, Febru[illegible] 1987.

[7] M. S. Horwedel et. al. *The Feature Extraction Processor: 1986 Progress*, Technical Information Series (unnumbered), General Electric, Valley Forge, PA, December 1986.

[8] *Geometric Arithmetic Parallel Processor*, NCR Product Specification NCR45CG72, Dayton, OH, 1984.

[9] *The Cross-Omega Connection Machine: A Multiprocessor System Architecture*, Proposal to DARPA, General Electric, Schenectady NY, November 1984.

IMAGE UNDERSTANDING TECHNOLOGY AND ITS TRANSITION TO MILITARY APPLICATIONS

David Y. Tseng
Artificial Intelligence Center
Hughes Research Laboratories
23901 Calabasas Road
Calabasas, CA 91302-1579

Julius F. Bogdanowicz
Electo-Optical and Data Systems Group
Hughes Aircraft Company
P.O. Box 902
El Segundo, CA 90245-0902

ABSTRACT

The various methods and levels of technology transfer, with the universities and within Hughes, are discussed. Examples of programs where effective technology tranfers took place are cited, including pertinent military applications of the image understanding technology. A case study of the Hughes image understanding program for photo-interpretation is presented in detail.

INTRODUCTION

Technology transfer takes place in several forms within Hughes and with the Universities, differentiated mainly by the state of the technology and its maturity for applications development. Those technologies that are in their embryonic stages, and needs fairly comprehensive research efforts to show feasibility, are generally initiated and pursued at the Research Laboratories. These projects inherently have a high risk for success, but are counter-balanced by the prospects of high payoff in being able to hurdle major technology barriers. Expectations of technology transfer to the operating divisions for these projects are usually in the 3 to 5 year time frame. The technology areas that are further along in their stages of development and can be expected to impact applications programs within 1 to 3 years, are usually intitiated jointly between the Research Laboratories and the operating divisions. Work on these programs is carried on by the two groups simultaneously, with the research aspects emphasized by the Research Laboratories and the applications aspects developed by the operating divisions. The third category of technology transfer usually involve programs underway at the operating divisions, where a specific area of technology need to be examined by the Research Laboratories to help overcome identified problems. These are initiated by the operating divisions, and the response is directed at a well focused technical area.

The interactions with university research for technology transfer are more focused toward the identified needs of the programs. Here, specific technologies will be utilized that have direct applicability to certain projects, whether it be research or prototype development. Generally, the transfer of technology is arranged through direct contact and coordination with the principal investigator of the university program.

A primary objective in pursuing technology development and its transfer into the operating divisions is to explore the usefulness of the technology in military application, with the eventual goal of incorporating them into actual systems to improve performance. In the past ten years, several significant technology areas have evolved in this way at Hughes. The common thread in these programs is the image understanding (IU) technology foundation from which several applications areas have been successfully explored. These programs include: The DARPA Autonomous Terminal Homing (ATH) program; the Navy Automatic Aimpoint Selection and Maintanence (AUASAM) program; the Army Advanced Tactical (ATAC) FLIR program; the Army Advanced Image Compression (AIC) program; the DARPA/ORD Image Understanding System (IUS) program; and the DARPA Knowledge-Based Vision Technology (KBVT). A brief summary of these programs is presented below, with the technology transfer mechnism and applications areas described, where appropriate. In addition, a more detailed case study of the DARPA SCORPIUS program is discussed as a successful example of technology transfer involving both internal Hughes organizations and universities.

SELECTED EXAMPLES

Autonomous Terminal Homing (ATH): The goal of the DARPA ATH program was to develop image-based techniques for use in mid-course navigational update and terminal homing of cruise missiles. This program involved multi-contractors, with the majority of the approaches utilizing variations of correlation-based techniques. Hughes Research Laboratories proposed a model-based technique which relied heavily on segmentation, line finding, and model building algorithms. In addition to developing the needed algorithms for use in the Hughes approach, results developed under various university IU programs were evaluated for incorporation into the Hughes system. The most useful one was the line finding algorithm developed by Nevatia and Babu at USC, and this formed a critical link in the model building process for the Hughes system. This technology was initiated and pursued at the Research Laboratories and transfered to one of the operating divisions of Hughes for bidding on the prototyping phase of the ATH program. Concurrently, a second approach was undertaken by one of the operating divisions, utilizing multi-level correlation techniques. This technique, together with a Fourier tranform method of classifying targets, had direct application in tracking targets and was used in such missiles as the maverick.

Automatic Aimpoint Selection and Maintanence (AUASAM): The Navy AUASAM program utilized the method of moments to select and track critically vulnerable areas of airborne targets for munitions delivery. Here, segmentation techniques were used to extract the silhouette of the target for use in moment calculations. These moment measures were then used to match against an extensive data base of pre-measured moments for a number of targets derived at a variety of look angles. Once matched, the approach angle of the target was known, and the vulnerable aimpoint on the target exposed. The system then tracks and maintains the aimpoint for munition delivery. This approach was developed at the Research Laboratories in response to an operating division request for assist on this program. Subsequently, a prototype system was built by the operating division and successfully tested by the Navy.

Advanced Tactical (ATAC) FLIR: The Army ATAC program involved the development of traget cueing techniques for automatic detection and recognition of tactical targets in infrared imagery. Because of the computationally intensive calculations required for segmentation, a preliminary processing stage was incorporated which selected interest points based on a number of simple statistical measures. Only regions in the neighborhood of these interest points were processed for segmentation of possible targets. Classification methods utilizing nearest neighbor classifiers provided identification of the targets. This program was initiated jointly by an operating division and the Research Laboratories, which worked closely on system development for 2 years. Prototype development then took place at the operating division, resulting in the design and development of the Electro-Optical Signal Processing Computer (EOSPC) to satisfy the real-time processing needs of the system. This work was the precurser to other automatic target recognition (ATR) programs, and was subsequently incorporation into the LANTIRN development.

Meanwhile, the efforts at the Research Laboratories were directed toward developing and utilizing AI approaches to target recognition. This shift in approach was prompted by our realization that the statistical, algorithmic approach to ATR had reached its limit of capability because of its inflexibility and inability to utilize scene context to improve the detection and recognition process. Under an Army program Investigation of Context Cueing (ICC), work has progressed during the past 4 years in developing an AI approach to ATR. In the current ICC system, both knowledge-driven and data-driven approaches have been incorporated into an integrated system. This system is presented at this workshop in an invited paper "Image Interpretation Using Scene Context".

Advanced Image Compression (AIC): Utilizing ATR and model-building IU techniques, extremely large bandwidth compression ratios were achieved in the Army AIC program for transmission of images. Here, compression ratios of up to 10,000:1 were attained for image transmission scenarios requiring severely reduced bandwidth. In this method, the compression technique preserved information content rather than overall picture quality. A hybrid scheme was used wherein the compressed image consisted of a range of resolutions and update rates: high priority targets were sent at full resolution in small windows, while lower priority targets were sent at lower resolutions, and the background scene was sent as an edge or line picture; the image update rates ranged from 6 frame per second for the high priority windows to 1 frame per second for the background scene. This program was intiated by the Research Laboratories, and was developed to the feasibility stage, at which time it was transfered to an operating division for prototype development. Subsequently, this technique was combined with target tracking techniques to form the Bandwidth Reduction and Intelligent Target Tracking (BRITT) program, which is currently under development at the operating division.

Image Understanding System (IUS): The many components of IU technology developed over the years were incorporated into the DARPA/ORD IU System program for automating the photo-interpretations process. This work evolved into the current SCORPIUS contract under the DARPA Strategic Computer Program, and will result in the development of research prototype analyst workstations. Details of this program are discussed in the case study.

Knowledge-Based Vision Technology (KBVT): Finally, a similar comprehensive utilization of IU technology is being undertaken for the program in Autonmous Land Vehicle (ALV) applications. The KBVT contract is also under the DARPA Strategic Computer Program, and the Hughes effort is focused on developing the perception system needed for obstacle detection and avoidance. Both on-road and cross country obstacles are of interest. In fact, any feature which poses a problem to the vehicle in terrain negotiation is considered as an obstacle. Because of the intimate interplay between planning and perception in the ALV, the development of these two systems is closely coordinated. Current status of this work is reported in the paper "Developments in Knowledge-Based Vision for Obstacle Detection and Avoidance", presented at this workshop.

CASE STUDY: SCORPIUS, AN IMAGE UNDERSTANDING SYSTEM FOR PHOTO-INTERPRETATION

This program serves as a good example of how technology transfer can be of benefit to both organizations involved in the transfer process, and how such coordination and cooperation provides the sponsoring agencies with a more productive program than either organization can produce alone.

The Ingredients: The initial requirements for participation in this program was well matched to the interests and capabilities of the two organizations: the Research Laboratories, which had the technology expertise and experience, and the Software Engineering Division of the Electro-Optical Data Systems Group (EDSG), which had the applications and systems expertise and experience. Furthermore, the problem of applying image understanding technology to photo-interpretation applications was of interest to both organizations. The fact that the goal of this program was aimed toward demonstrating the feasibility of this technology and not in building operational systems, further contributed to the overall appeal. The close ties which the Research Laboratories had with the university community added to the program structure. There were strong commitments from the management of both organizations to have the group work together as a close-knit team. (The program manager at EDSG had formerly been the manager of the Research Laboratories group). Finally, a small team of highly motivated individuals added to the success of this program.

The Approach: The proposal was a joint effort, with the technology and approach being the primary responsibility of the Research Laboratories and the system design and program management being the primary responsibility of the Software Engineering Division of EDSG. It was decided that the

ACRONYM vision system developed by Rod Brooks at Stanford University would be used as the core of the proposed IU system. This decision was based on the fact that ACRONYM was the most comprehensive and integrated vision system in existence at that time, and it encompassed many of the key research issues identified for the program. In the first phase of the program, the issues relating to the transfer of ACRONYM into the selected photo-interpretations domain were primarily pursued by the Research Laboratories, with consulting help from Brooks. The system integration, and implementation were done by the Software Engineering Division. Frequent and detailed program reviews took place in order to keep the sponsoring agencies up-to-date on program progress and technical barriers encountered. The fact that the COTR was not only keenly interested in the program, but also highly competent technically, was critical to the direction which the program evolved and the progress made.

Phase I Findings: During this phase the participation of the two Hughes organizations was split nearly 50% each. A major portion of the research was devoted to enhancing or modifying certain of the limitations in ACRONYM for use in the photo-interpretations problem. These limitations included the lack of shadow prediction capability; object identification based only on shape information; modeling of objects by simple generalized cylinders only; and prediction capability tested only for a vertical camera position. Working closely with Brooks, who was a consultant on this program, the ACRONYM system was modified to improve some of these capabilities, while other problems remained.

Phase IIa,b Findings: The process of technology transfer had been occuring throughout phase I, and at this point the program entered a more applications oriented stage. The participation of the Research Laboratories leveled off to about 30%, with the bulk of the program being developed by the Software Engineering Division. Our experience with ACRONYM convinced us that the model driven predictive approach was going in the right direction. However, the limitations in the constraint manipulation theory and implementation language caused us to redirect our approach. Rather than continuing further modifications and development of ACRONYM, the technical approach shifted toward a graphics-based prediction method with a symbolic interpretation module. The graphics modeling program, MOVIE.BYU, was obtained from BYU and the image processing software, GIPSY, was obtained from R. Haralick at VPI. Both of these were integrated into the phase II system and have continued to evolve to provide the basis of the SCORPIUS phase. A symbolic interpretation module was developed during this phase of the program. At the time, no frame-based shell for reasoning with heuristics was available for the VAX, so a rule and frame system (ARF) was developed at the Research Labs. A post-processing package, which was also developed at the Research Labs, was required to interface the MOVIE numeric predictions with symbolic representations required in ARF. As a result of this work, an improved and better integrated system was defined for the prediction and interpretation modules for the SCORPIUS phase. In this, ARF has been replaced by MOBIUS, which is being implemented by Software Engineering Division, to upgrade the frame system, provide foward and backward chaining, and interactive programming and debugging tools. In addition, BYU has incorporated the symbolic prediction capabilities directly into their MOVIE package.

SCORPIUS Phase: As this phase of the program started, the scope and emphasis shifted dramatically. The scope of the program increased 10-fold, and is focused on the development of an experimental prototype analyst workstation using the hardware and software developed under the DARPA Strategic Computing Program. The emphasis is on application development, resulting in a prototype workstation. Because of this, the role of the Research Laboratories is now that of consultation. In phase II of the program, we found that shape matching is not sufficient to recognized all objects of interest. We believe that constraint based reasoning is an important part of a capable vision system architecture. Along this line, the constraint satisfaction work of J. Pearl at UCLA has been incorporated into this system. This phase of the program also entered another significant stage by incorporating the experimental hardware (the BBN Butterfly and CMU WARP processors) and the associated operating environments developed in the Strategic Computing Program.

Phase IIc: As the IU System program evolved toward the applications emphasis, this phase of the program was started in order to provide the continuity necessary to meet the research needs of the program in the future. The focus now is on the development of advanced object detection techniques. As new techniques are developed, they will be incorporated into the more applications oriented systems and workstations for evaluation and fine tuning. In this phase, the Research Laboratories is again participating at a 50% level and guiding the research to meet the future needs of this program. This program is just beginning and a review of recent research in the IU community will be facilitated with the consulting aid of Dr. Rosenfeld.

Retrospective: The process of technology transfer and interactive development of joint programs requires determination and discipline. Although very significant progress can be made, there are many factors which make it difficult to achieve a smooth transition. For instance, even mundane issues such as the large distance separating organizations participating in technology transfer makes it difficult to interact frequently, and necessitates the duplication of testbeds at the facilities. However, when the transfer of technology does take place, the benefits are many and quite rewarding. Technology can, and does, migrate from university to industry. Our experiences have found that the research systems in universities are typically not developed to a stage that supports direct application into operating military systems; instead, additional research by Hughes together with the consultation and close interaction with the university is required. Research software systems (either from universities or industrial research laboratories) are usually not designed to be complete packages encompassing the needs of application systems. The operating divisions do benefit tremendously from the Research Laboratories by adopting new concepts, approaches and results. The Reserach Laboratories, in return, benefit greatly by its exposure to the complexities of real-world problems, and gain an appreciation of the system aspects of a problem. In summary, technology transfer may be difficult, but it can work very well if the right ingredients are in place. Both management and program personnel must be firmly commited to make it happen. Technology transfer occurs in both directions, with each organization benefiting from the strength of the other.

Context Dependent Target Recognition

Teresa M. Silberberg

Hughes Artificial Intelligence Center
23901 Calabasas Road
Calabasas, CA 91302

Abstract

In this paper we describe the Investigation in Context Cueing (ICC) system which incorporates scene knowledge in a target detection system thereby providing a more reliable classification of the objects that appear in the images. Scene knowledge consists of models of objects expected in the scene and their relationships as well as information related to image acquisition. Extracted image features represented as symbolic descriptions and a network of scene object models described using frames drive the interpretation in both a bottom-up and top-down fashion. The interpretation process is object-oriented, that is, each hypothesized scene object independently gathers information which provides evidence for or against the hypothesis. The system has been exercised on a number of forward looking infrared (FLIR) images and has exhibited performance which exceeds that of traditional approaches.

We briefly discuss the symbolic representation of the image features and the network representation of the object models. The processes of model instantiation and evidence gathering and evaluation are then described in some detail. We discuss the current knowledge base, and lastly, we demonstrate the system incorporating target formation and motion knowledge by way of two examples.

1. Introduction

An image interpretation system has as its objective the identification of expected scene objects in an image. Most image interpretation systems incorporate the use of scene knowledge at one or more stages of the interpretation process. The need for this incorporation stems from the fact that real world scenes are very complicated, conditions under which images are acquired often lead to highly variable appearances of known scene objects, and generic image processing often leads to features that have no counterparts in the real world. Scene knowledge typically consists of models of objects expected in the scene and their relationships as well as information related to image acquisition. Knowledge may include, for example, whether trees and roads or desks and chairs are to be expected as well as object descriptions and spatial and intensity relationships. Additionally, knowledge may include ancillary information such as the lighting conditions, sensor type and sensor parameters. A third type of knowledge which systems are beginning to incorporate is expected image feature types, derivable from object characteristics, such as antiparallel line pairs or blobs of specific size and brightness.

The process by which areas in an image are interpreted as scene objects can be broken into several modules. Communication and feedback between selected modules as well as infusion of scene knowledge into appropriate modules has the effect of improving the robustness of the interpretation process. A simple example of an interpretation system is shown in Figure 1. In this example there are two modules: the Symbolic Description Module and the Interpretation Module. The Symbolic Description Module transforms the original image into symbolic descriptions using generic image processing algorithms as well as specialized algorithms that extract expected features types. Any of the algorithms may use ancillary information to refine parameter values, for example, the time of day can have an effect on the degree of contrast in the image and hence be used to guide an intensity rescaling procedure. The Interpretation Module applies the ancillary information and the object models to the symbolic descriptions in order to yield a consistent description of the scene. These two modules interact until the best interpretation is found.

One can think of the Symbolic Description Module as consisting of a pool of both general and specific image processing algorithms. Some algorithms may be invoked in a predetermined manner, for example, typical preprocessing algorithms include intensity rescaling and image smoothing; on the other hand, some algorithms will be applied after consideration of ancillary information, expected feature types, and more importantly, as a result of findings by the Interpretation Module. In a system by Nevatia [13], for example, long antiparallel lines, strongly indicative of runways, are found as a step in constructing a map of an airport complex. The Interpretation Module can usually be thought of as either a rule-based system in which the scene knowledge has been represented as production rules or an object oriented system in which the scene knowledge is represented as procedures attached to specific objects.

The use of expected features types in the segmentation process can be seen in the region analysis of Yakimovsky and Feldman [20]. Beginning with an initial partition of the image, pairs of regions are iteratively merged if they have the weakest boundary characterized by properties such as the relative size and intensity of the regions and the boundary length. In the rule-based system of Nazif and Levine [12, 9], both region and line properties are utilized to segment an image. The rules determine how an initial set of regions and lines are modified (for example, split or merged for regions and deleted or extended for lines) to yield uniform regions and connected lines.

Several systems incorporate scene knowledge to interpret an image. McKeown et al. [10] propose a rule-based system which uses airport models to improve an initial segmentation. The interpretation process involves building a complete airport instance in an incremental fashion while checking for consistency. As a specific interpretation evolves, the segmentation is modified to provide additional consistent hypotheses. The ACRONYM system (Brooks [2]) formulates general classes of three-dimensional objects and uses both geometric and symbolic reasoning to identify aircraft in aerial photographs. The interpretation process which determines an object's orientation and position is iterative. Initial predictions of objects, based on invariant features of the image, generate structural and spatial constraints on the location of the object. As additional matches are found, the details of the predicted features become finer and the location of the object becomes more constrained. VISIONS (Hanson and Riseman [5]) uses a hierarchical model representation to guide the interpretation of multispectral images of outdoor scenes. A scene is modelled as a schema which specifies common objects in the scene and their relationships. The model hierarchy includes volumes and surfaces that make up an object in a particular schema. Bottom-up and top-down processing match image features to the hierarchy of models to construct an interpretation. Binford [1] presents a good survey of model-based image understanding systems.

In this paper we describe an image understanding system with the architecture of the system in Figure 1. Effort has been concentrated on the formulation of spatial and temporal knowledge and the development of the Interpretation Module which utilizes an object oriented approach with models described using frames and a semantic network. Since the focus of this system has not been on developing and incorporating an intelligent segmentation stage, only a limited number of low level image analysis algorithms have been implemented. The system has been exercised on a number of forward looking infrared (FLIR) images.

In Section 2 an overview of the system architecture is provided. The current knowledge base as well as the performance of the system on two images are presented in Section 3. Lastly, Section 4 discusses extensions and concluding remarks.

2. System description

This section describes scene knowledge representation and the way an image is interpreted or *classified*. An effective structure for representing image-based or iconic data must support operations that retrieve pixel properties such as intensity, edge magnitude and what objects exist at a pixel; object properties such as area, compactness and direction of motion for regions or length and orientation for lines; and object spatial relationships such as adjacency and nearness. The Symbolic Pixel Array (Payton [15]) which allows efficient retrieval of pixel and object properties and object spatial relationships is used. Each pixel property is stored as an array, and an object is stored as a binary mask with x and y offsets. Object properties are computed as they are needed and then stored explicitly with the binary mask. By performing logical operations on the masks spatial relationships can be computed quickly.

First, object representation using frames with slots and object classification using evidence gathering and evaluation is described. Next, the organization of the scene knowledge in a classification structure and the use of this structure as a guide in the Interpretation Stage is presented.

2.1 Evidence gathering and evaluation

The classification approach is object-oriented, that is, each symbolic description is initially hypothesized or *instantiated* as one of the scene objects. The instantiated object then gathers information which will provide evidence for or against the hypothesis. The scene knowledge, represented in the form of object models and associated rules, directs the gathering of the evidence. If enough evidence is found, the hypothesis then becomes *established*, and the degree to which all of the evidence confirms or disconfirms the established hypothesis is computed. Since the original symbolic descriptions are highly dependent on the image processing algorithms used to extract them, evidence will necessarily contain some uncertainty. A means of correctly combining this uncertainty at any stage in the processing is an integral part of any image interpretation system. Figure 2 indicates the specification of an object model and demonstrates how the model is instantiated.

Each object model is characterized by attributes and desirable relationships between these attributes. The presence or absence of each attribute provides evidence that confirms or disconfirms an instantiation of that object. Attributes and their relationships are represented in the model by way of *slots* and *rules* that specify restrictions and relationships between these slots. Referring to Figure 2, a slot is specified by four pieces of information: the *role* specifies the kind of scene object that may fill the slot; the *quantity* restricts the number of objects that may fill the slot; the *acceptability* indicates the minimum confidence level the slot candidate must have; and the *seeker* provides means of filling the slot. An object to fill the slot is found by the *finder* if such an object exists; otherwise the *hypothesizer* tries to hypothesize an appropriate object. The last element of a seeker is the *screener* which screens out any slot candidate that is clearly unsuitable. The *restriction and relationship rules*, formulated as Rejection, Validation and Combination rules, represent a collection of information that (1) determines if a symbolic description should be instantiated as a particular scene object, (2) specifies exactly which slots must be filled, and (3) describes how to combine the evidence so as to compute an overall confidence.

Again referring to Figure 2, the instantiation and evidence gathering process is described in more detail. A scene object, if it is not rejected by the Rejection rule of an object model, is instantiated as a *hypothesis*. A "blackboard" is used to contain all hypotheses. The seeker of each model slot instantiates a *seeker instance* which invokes its associated procedures. These procedures provide *possibilities* for the slots. When the slots are filled according to the Validation rule, the hypothesis then becomes *established*. At this time actual *values* for the slots must be chosen from the possibilities. That is, if the slot quantity specifies that exactly two values are allowed but six are possible, the "best" two must be chosen. To do this, the Combination rules are evaluated for each combination of possibilities, and that combination which results in the highest confidence is considered "best". The evaluation and combination of uncertain evidence follows the method set forth in MYCIN [3], that is, we maintain an overall confidence measure computed from a measure of belief and a measure of disbelief. It is important to note that if a new object which passes a slot screener is added to the blackboard, the Combination rules are evaluated again causing the slot values to be reconsidered.

2.2 The classification process

In the classification process, it is desirable to withhold committment to a classification if insufficient evidence is present and to avoid complex analysis of an incorrect classification. Additionally, rather than attempting classification of a symbolic description directly as an expected object, it is often more appropriate to gather pieces of evidence which can then hypothesize that object. A general-to-specific hierarchy can be utilized in realizing the former objective; a model that specifies composite pieces can be helpful in realizing the latter.

The structure used in representing the scene knowledge is a semantic network which is a directed graph consisting of a set of vertices and a set of labelled edges between pairs of vertices. In this representation, a vertex is an object type, and the label on an edge represents either an "a-kind-of" or an "a-part-of" relationship. Figure 3 shows the representation of objects in the current scene model. The solid edges represent the "a-kind-of" relationship ("line-road is a-kind-of line-scene-object"). Notice that every object is part of the "a-kind-of" hierarchy. Broken edges represent "a-part-of" relationships ("vehicle is a-part-of formation") and also possible object dependencies where the object at the tail of the edge confirms the object at the head (vehicle confirms road). We include vehicle-parts to demonstrate how more complicated relationships can be incorporated into the network. Here, the confirming evidence of windshield and wheel for vehicle is inherent in the edge from vehicle-parts to vehicle.

A summary of the classification process is depicted in Figure 4. Initially each of the original symbolic descriptions is hypothesized as the most general object class (scene-object in Figure 3). For each hypothesis, the associated seekers are instantiated and either find known objects or hypothesize new objects to fill the object slots. In the event that one or more objects hypothesize another object, the "a-part-of" relationships are utilized. Objects that are "a-part-of" another can singly or jointly hypothesize that other object (wheel and/or windshield can hypothesize vehicle); furthermore, any object and its "parts" can jointly hypothesize another object (a road and a vehicle not "on" a road can hypothesize a new road which is "underneath" that vehicle).

When reasonable evidence has been supplied to all of the slots of a hypothesized object, that object becomes established, and slot values that yield the highest confidence are chosen from the possibilities list. For each established hypothesis, a subclassification following the "a-kind-of" edges backwards proceeds unless a base class has been reached. Once all new hypotheses have passed their respective Rejection rules, the classification process begins again. The classification loop continues until no new hypotheses are generated.

Before the final classification of an object can be determined, the confidence computed for a specific class must be spread throughout the whole "a-kind-of" hierarchy. In other words, confirming evidence for one class should have the effect of confirming each class in the path from the root to that class and disconfirming every other class. For example, referring to Figure 3, confirming evidence for vehicle should likewise confirm region-scene-object and scene-object and disconfirm every other class (line-scene-object, line-road, road-subregion and so on). Similarly, disconfirming evidence for one class should also disconfirm every subclass of that class and confirm every other class. The final classification of an object results in that class with the strongest global evidence provided that the evidence is above some preset threshold.

The current technique for global confidence computation distributes the evidence only partially over the "a-kind-of" hierarchy in that confirming evidence for a class confirms only its superclasses without disconfirming unrelated classes; furthermore, disconfirming evidence for a class disconfirms its subclasses without having a confirming effect on other classes. See Kim [8] for details. There are several other techniques for combining evidence over a hierarchy. These include techniques based on the Dempster-Shafer theory of evidence (Shafer [19], Gordon and Shortliffe [4]) and techniques based on Bayesian theory (Pearl [16]). Both techniques, one of which may be incorporated at a future time, are attractive since they are mathematically tractable and reasoning about sets of classes is possible.

3. Experimental results

In this section the results of classifying several images using a specific model and an implemention on a Symbolics 3640 are presented. In Figure 5a an image which is typical of those to which the model has been applied is shown. The thinned and thresholded edges and line segments computed by applying the line finder of Nevatia and Babu [14] are in Figures 5b and 5c, respectively. Figure 5d shows the segmented image resulting from an edge-based segmentation (Perkins [17]). Before the classification results are presented, the knowledge base is described.

3.1 System knowledge base

Since the current images were all taken by a FLIR sensor from high altitude, the knowledge base does not explicitly include ancillary information such as sensor type, time of year, and sensor distance. Apart from the roads and vehicles, there is very little other information, thus making most ancillary data irrelevant. The object network is similar to that in Figure 3 except VEHICLE-PARTS, WINDSHIELD and WHEEL are not included. Since the models for FORMATION (**L**, **W**) are relatively complicated, they are discussed in some detail after briefly discussing the other object models. This subsection is concluded with a description of a simple technique whereby moving vehicles are identified through the analysis of sequences of images.

Obviously, SCENE-OBJECT is the general class for both region and line segments, and LINE-SCENE-OBJECT and REGION-SCENE-OBJECT are general classes for line segments and regions, respectively. LINE-ROAD is used to allow a line segment of a certain minimum length to hypothesize a road on both sides of it if there is not one already. ROAD-SUBREGION corresponds to a small, elongated region which is part of a road that has been segmented into pieces. A road-subregion looks for line-scene-objects and roads in nearly the same orientation, and it is able to hypothesize a road. VEHICLE, a small, compact and bright region, is usually hypothesized by either ROAD, FORMATION or evidence of a moving vehicle.

A road object must be large, elongated and a different intensity from its neighbors. In this model, evidence for a road consists of strong, parallel lines, vehicles, and aligned and nearby subregions. ROAD is frequently hypothesized by ROAD-SUBREGION or LINE-ROAD since roads are often segmented into smaller regions, and lines on at least one side of the road are often present. A road may also be hypothesized by a vehicle not yet "on" a road and a nearby road in the correct orientation. As we show in an example, the segmentation algorithm will not necessarily yield regions which correspond to vehicles; thus an important task of a road is to hypothesize a vehicle. According to the Validation rule, in the event that there is evidence of a joining road, not all road slots need to be filled; furthermore, if the line-road slot is filled and either the road-subregion or the vehicle slot is filled, then the other slots do not need to be filled.

3.1.1 Inclusion of formation knowledge

Since military vehicles frequently appear in groups, formation knowledge can be used to predict the appearances of undetected vehicles and to increase the confidence of vehicles that appear in formations. We currently consider two kinds of formations: an L (linear) formation in which at least three vehicles are regularly spaced in a relatively straight line, and a W formation in which the arrangement of five vehicles forms a "W". An important assumption for W formations is that distinct formations are not close to one another relative to the size of the formation; more specifically, if a vehicle near a formation does not belong to that formation, then it will not belong to any formation. We discuss (1) how vehicles find and hypothesize formations, (2) how formations hypothesize vehicles, and (3) the effectiveness of the formation building techniques.

3.1.1.1 Formation hypothesis from vehicles

Formations are built incrementally as vehicle hypotheses become established. The general strategy for building L and W formations is the same. The first of a formation type is created by any two vehicles satisfying some criteria $C_{2\text{-vehicles}}$. A vehicle not yet belonging to a formation adds itself to an existing formation if additional and possibly more restrictive criteria $C_{\text{add-on}}$ are met. Finally, if a vehicle is unable to add itself to an existing formation, one or more formations are created as long as $C_{2\text{-vehicles}}$ are met. In the case of an L formation, new formations consist of the vehicle and one of each of the vehicles already in a formation; in the case of a W formation, the new formation consists of the vehicle and the nearest other vehicle not yet in a formation.

$C_{2\text{-vehicles}}$ for L formations and W formations require that the two vehicles be nearby. $C_{\text{add-on}}$ for L formations require that the vehicle is near the formation and that the resulting formation is still relatively linear; $C_{\text{add-on}}$ for W formations require only that the vehicle is near the formation. The most consistent arrangement of vehicles in a formation cannot be known until all vehicles contributing to that formation are found; thus, while formations are being built, no attempt is made to enforce strict spatial arrangement constraints. By construction, an L formation consists of nearby vehicles arranged linearly, and a W formation consists of a cluster of nearby vehicles.

3.1.1.2 Vehicle hypothesis from formations

The model of a formation has only a vehicle slot whose possibilities list is initially filled by the vehicles that created the formation. Additional vehicles are included if they have added themselves to the formation or if they are hypothesized by the formation. L formations hypothesize vehicles by looking for a relatively bright area in the original image along a line which is in the direction of the formation's primary axis, has the same center as the formation, and has length three times longer than the formation. Notice that this technique does not attempt to hypothesize vehicles at regularly spaced intervals because the correct spacing is not actually known until all possible vehicles are available to the formation.

The procedure that W formations use to hypothesize vehicles is more complicated. A two-dimensional model of a W formation is known in advance. Given a hypothesized W formation (a cluster of vehicles), possible transformations (scale, translation and rotation) of the model to the formation are computed by matching the vehicles pairwise. Correspondences to the transformed model are determined by examining the vehicles in the hypothesized W formation as well as further examining the image for a possible vehicle. A goodness of fit measure associated with the match characterizes the tradeoff between the number of vehicles in correspondence and the sum of the distances between corresponding vehicles. The match with the highest goodness of fit is then used to hypothesize new vehicles in those cases where evidence of a vehicle was found after examination of the image.

3.1.1.3 Effectiveness of formation hypothesis

The formation building strategy is considered effective if all of the correct formations are found with high confidence and if incorrect formations have low confidences. In the case of L formations, all correct formations are found since, by construction, if a vehicle cannot add itself to an existing formation, then it creates new formations containing itself and one of each of the vehicles in a formation. Because L formations are not perfectly linear, all correct formations will be formed up to "transitivity" where two formations overlap and should be considered as one. This can be resolved by a formation combination process. Several incorrect formations will also be formed; it will be the responsibility of the Combination rules to insure that those formations having less than three vehicles, a low degree of linearity or a low degree of spatial regularity will have lower confidence than those that are correct.

Recall that for W formations it was assumed that distinct formations are located far apart. Since hypothesized W formations are essentially clusters of nearby vehicles, we restrict "nearby" to reflect a combination of the farthest distance between two vehicles in a W formation and the closest distance between two W formations. Since a vehicle in a cluster belongs either to that W formation or to no formation at all, and since the best match to the W formation model is determined during the evidence evaluation phase, all correct W formations will be found and incorrect W formations will have low confidences.

3.1.2 Motion analysis

A strong indication of a vehicle is its motion from frame to frame. This motion can be used to predict or support a vehicle hypothesis. In this section we describe a simple technique whereby the location of a vehicle that has moved between frames is detected with respect to the first frame. While interpreting the first frame, if a vehicle does not yet exist where a moving vehicle is expected, it is hypothesized.

Starting with two registered frames in a sequence, a difference image is computed by subtracting one from the other, pixel by pixel. (See Jain and Nagel [7] and Jain et al. [6] for work on difference images.) Ideally, a difference image would have pairs of negative and positive regions denoting where space has been vacated and filled by each moving object. In general, the resulting difference image is thresholded using a multiple of the standard deviation of the difference image values which approximates the standard deviation of the sensor noise. Those pixels that have an absolute difference less than the threshold are set to zero.

Given a difference image with nearby negative and positive regions, the next step is to determine where the moving vehicles are located in the first frame. Since the location of the moving vehicle in the first frame corresponds in part to the vacated area in the difference image, segmentation of that area in the first frame will yield a reliable region which corresponds to the vehicle. For the images on which the motion analysis has been performed, a simple region growing technique has been used.

To take full advantage of multiframe analysis, the interpretation of one frame can be "projected" into the interpretation of the next frame. For example, areas where roads have been hypothesized in one frame can indicate where in the following frame they can be expected. Likewise, the location of a moving vehicle with an associated direction and velocity can be projected into the following frame. A complete interpretation history would include the interpretation results from each preceding frame. One alternative to such a space consuming history involves a more sophisticated evidence combination scheme. This scheme would entail tagging a hypothesized object if it is not found in a successive frame. The confidence of this tagged object would decrease as it is not found in successive frames. The amount of the decrease may be dependent on the object type or the amount of storage available. Eventually, unless the object is found in a successive frame, it is removed from the interpretation and will no longer be "projected". For example, if a vehicle is found in one frame but occluded in the next, it is tagged and its confidence is reduced. For each successive frame in which it is occluded, its confidence is reduced. If it is found again before its confidence is too low, it is untagged, otherwise it is removed from the interpretation. This technique for "projecting" an interpretation and decreasing the confidence for a tagged hypothesis is still being formulated.

3.2 Examples

We now present two examples to demonstrate how vehicles are found using road, formation and motion information. For brevity, an example that incorporates W formations is omitted. The classification structure is the same as that shown in Figure 3 except that vehicle-parts, windshield and wheel are not included. In all of the examples, line and region segments are extracted as described earlier (Nevatia and Babu [14] and Perkins [17]).

The data for the first example was displayed in Figure 5. The original image, thinned and thresholded edges, line data, and region borders overlayed onto the original image are shown in Figures 5a-d, respectively. There is one road with high intensity and varying width and eight bright targets positioned on either side of the curve. The lower part of the road is not well defined by line segments; furthermore, regions just below the curve in the road are not in the direction of the road. Only two of the eight vehicles have corresponding regions in the segmentation. Based on linear formation information, two formations are expected; one nearly vertical, the other horizontal.

The resulting classification is shown in Figures 5e and 5f. Figure 5e shows the road (light gray), road-subregion (dark gray) and line-road (white) objects. Both road-subregions and line-roads have hypothesized roads. Several road objects are at the wrong orientation due to the orientations of the road-subregion objects that hypothesized them. Figure 5f shows the vehicle objects (white and dark gray) on the road objects (light gray). The white vehicles on the upper part of the road belong to one linear formation, the dark gray to another. The hypothesized vehicles resulted from both road objects and L formation objects. Several vehicles were incorrectly hypothesized due to the sensitivity of the vehicle hypothesizer. If motion information were incorporated into the classification process, this ambiguity would be resolved.

In the second example, Figure 6, we show how motion analysis can be used to improve classification. We analyze the first frame incorporating the difference image analysis of the first and second frames. The two frames are shown in Figures 6a and 6b. These are synthetic FLIR images from a terrain board. There is a wide river and a road to its right winding from the middle left, down the center of the image. On the upper, horizontal part of the road, there are eight bright moving vehicles. Only three or four are clearly apparent, chiefly due to their bright intensity and regular spacing. There is another road without any vehicles that enters from the lower left and approaches the river. The thinned and thresholded edges of the first frame, line data, and region borders overlayed onto the first frame are shown in Figures 6c-e, respectively. The upper, horizontal part of the road along the river is poorly segmented, and only one of the eight vehicles is segmented.

The thresholded difference image in which black is a negative difference and white a positive is in Figure 6f. The difference threshold chosen is ten which is about five standard deviations of the difference image values. There are seven negative/positive pairs in the difference image. The second and third pair from the left are close to where a vehicle is expected but seem to result from noise. For each of these pairs, we "grow" a vehicle region in the first frame starting with the centroid of the negative region (vacated space) thereby finding the location of a moving vehicle in the first frame. Figure 6g shows the road, road-subregion and line-road objects resulting from the classification process. The upper part of the river road was not found very well; the left center road was found. Figure 6h shows the vehicle objects on the road objects. Six correct vehicles were found along the upper road almost entirely due to the moving vehicle knowledge which enabled five of the correct vehicles to be hypothesized. Several incorrect vehicles were found on the lower road, and again, with the addition of more extensive multiframe analysis, the incorrect vehicle hypotheses would be removed.

4. Conclusions

We have described an image interpretation system that efficiently represents symbolic image descriptions and scene context and effectively applies the scene context to the symbolic descriptions. All symbolic descriptions, including the hypothesized objects, are represented in the Symbolic Pixel Array which allows uniform treatment of all image-based objects in the system. The system incorporates a model representation that includes a hierarchy that is general enough to be easily extended to include additional objects. The simple interpretation process adhers to the principle of least committment in two ways: (1) using the model "a-kind-of" relations, object hypotheses occur only if there exist supporting intrinsic feature properties, and (2) final interpretations are not determined until all hypotheses have been made. The system incorporates both data and model driven processing to achieve performance that exceeds that of traditional approaches. Finally, spatial and temporal knowledge in the form of formations and simple motion analysis has been shown to improve the likelihood of correctly finding vehicles.

There are several extensions that will make the system more general. Since the extracted image features are of poor quality, weak evidence must be included in the object models thereby making the models sensitive to changes in image features. The inclusion of knowledge-based image processing algorithms which yield regions that more closely resemble scene objects would help alleviate this deficiency. The object models are currently in terms of their expected appearance in the image whereas a more general system would include three-dimensional shape information such as sticks, plates and blobs to define rough descriptions (Mulgaonkar et al. [11]) or generalized cylinders to define more specific descriptions (Brooks [2]). The incorporation of more extensive multiframe analysis would also be a desirable extension. Finally, it is possible to decrease processing time by implementing the current system using a parallel processor such as the Cosmic Cube (Seitz [18]) in which all objects that correspond to a particular symbolic description are allocated to a single processor, and message passing capabilities are utilized in the evidence gathering process.

Acknowledgements: The support of the Night Vision Laboratory under Contract DAAK21-84-C-0131 is gratefully acknowledged as is the assistance of Karen E. Olin, David W. Payton and others in the Hughes Artificial Intelligence Center for laying the ground work and providing strong support during the ongoing system development.

References

[1] T. Binford, Survey of model-based image analysis systems, *International Journal of Robotics Research* **1**, 1, pp. 18-64, 1982.

[2] R.A. Brooks, Symbolic reasoning among 3-D models and 2-D images, *Artificial Intelligence* **17**, pp. 285-348, 1981.

[3] B.G. Buchanan and E.H. Shortliffe, eds., *Rule-based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project*, Addison-Wesley, Reading, Massachusetts, 1984.

[4] J. Gordon and E.H. Shortliffe, A method for managing evidential reasoning in a hierarchical hypothesis space, *Artificial Intelligence* 26, pp. 323-357, 1985.

[5] A.R. Hanson and E.M. Riseman, VISIONS: A computer system for interpreting scenes, pp. 303-333 in *Computer Vision Systems*, eds. A.R. Hanson and E.M. Riseman, Academic Press, Orlando, Florida, 1978.

[6] R. Jain, W.N. Martin and J.K. Aggarwal, Segmentation through the detection of changes due to motion, *Computer Graphics and Image Processing* **11**, pp. 13-34, 1979.

[7] R. Jain and H.-H. Nagel, On the analysis of accumulative difference pictures from image sequences of real world scenes, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **1**, pp. 206-214, 1979.

[8] J.H. Kim, A distributed computational model of plausible classification reasoning, *Proceedings of the Second Conference on Artificial Intelligence Applications*, Miami Beach, Florida, pp. 210-214, 1985.

[9] M.D. Levine and A.M. Nazif, Rule-based image segmentation: A dynamic control strategy approach, *Computer Vision, Graphics and Image Processing* **32**, pp. 104-126, 1985.

[10] D.M. McKeown, Jr., W.A. Harvey, Jr., J. McDermott, Rule-based interpretation of aerial imagery, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 7, pp. 570-585, 1985.

[11] P.G. Mulgaonkar, L.G. Shapiro, R.M. Haralick, Recognizing three-dimensional objects from single perspective views using geometric and relational reasoning, *Proceedings of the IEEE Conference on Pattern Recognition and Image Processing*, Las Vegas, Nevada, pp. 479-484, 1982.

[12] A.M. Nazif and M.D. Levine, Low level segmentation: An expert system, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **6**, pp. 555-577, 1984.

[13] R. Nevatia, Image understanding research at USC: 1984-85, *Proceedings of DARPA Image Understanding Workshop*, Miami Beach, Florida, pp. 10-15, 1985.

[14] R. Nevatia and K.R. Babu, Linear feature extraction and description, *Computer Graphics and Image Understanding* **13**, pp. 257-269, 1980.

[15] D.W. Payton, A symbolic pixel array representation of spatial knowledge, *Proceedings of the IEEE Conference on Computers and Communications*, Phoenix, Arizona, pp. 11-16, 1984.

[16] J. Pearl, On evidential reasoning in a hierarchy of hypotheses, *Artificial Intelligence* 28, pp. 9-15, 1986.

[17] W.A. Perkins, Area segmentation of images using edge points, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2, pp. 8-15, 1980.

[18] C.L. Seitz, The Cosmic Cube, *Communications of the ACM* 28, pp. 22-33, 1985.

[19] G. Shafer, Hierarchical evidence, *Proceedings of the Second Conference on Artificial Intelligence Applications*, Miami Beach, Florida, pp. 16-21, 1985.

[20] Y. Yakimovsky and J.A. Feldman, A semantic-based decision theory region analyzer, *Proceedings of the International Joint Conference on Artificial Intelligence*, Stanford, California, pp. 580-588, 1973.

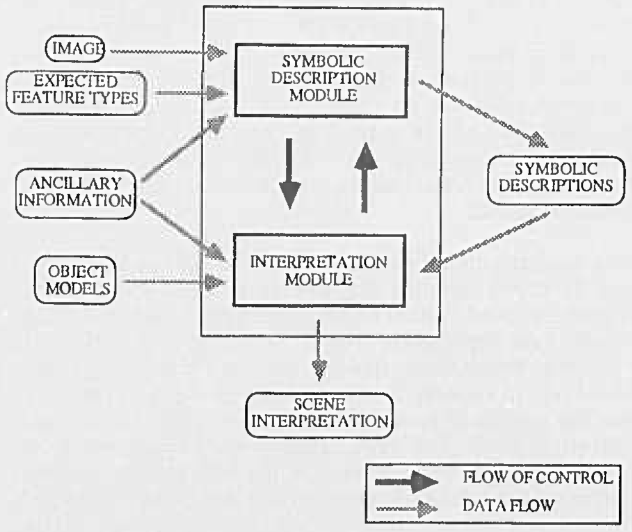


Figure 1. An image interpretation system.

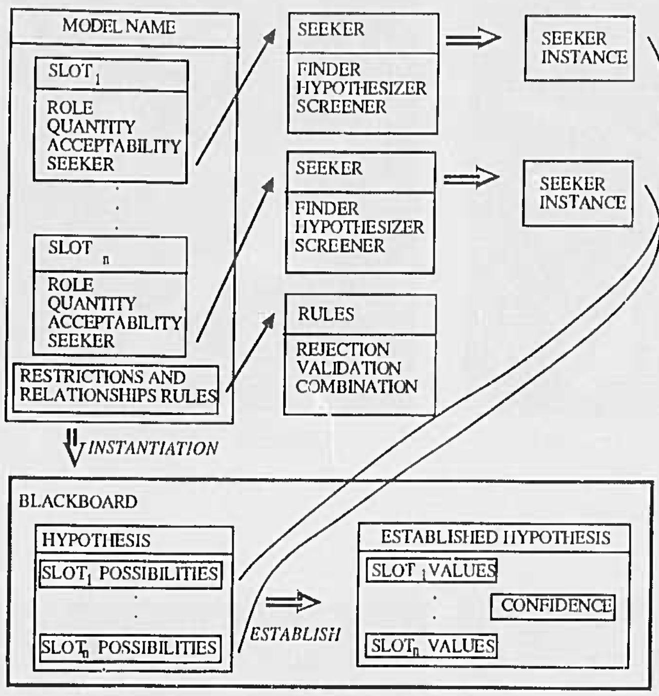


Figure 2. Instantiation and evidence gathering.

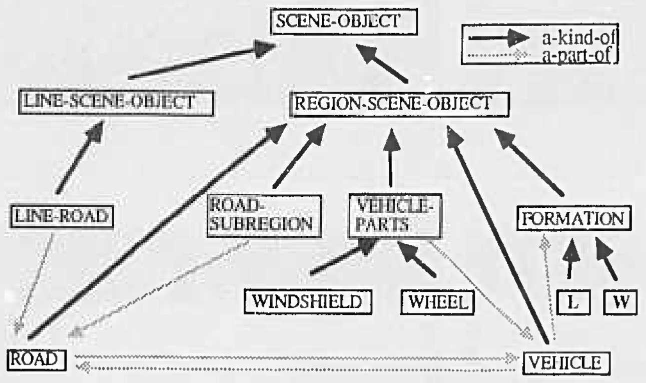


Figure 3. Semantic network representation of the scene objects.

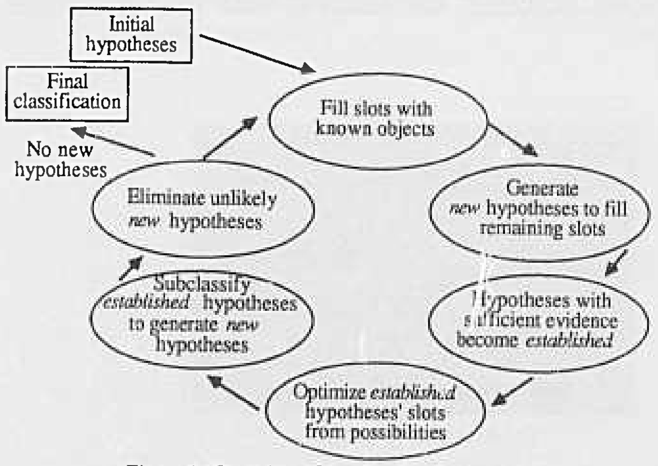
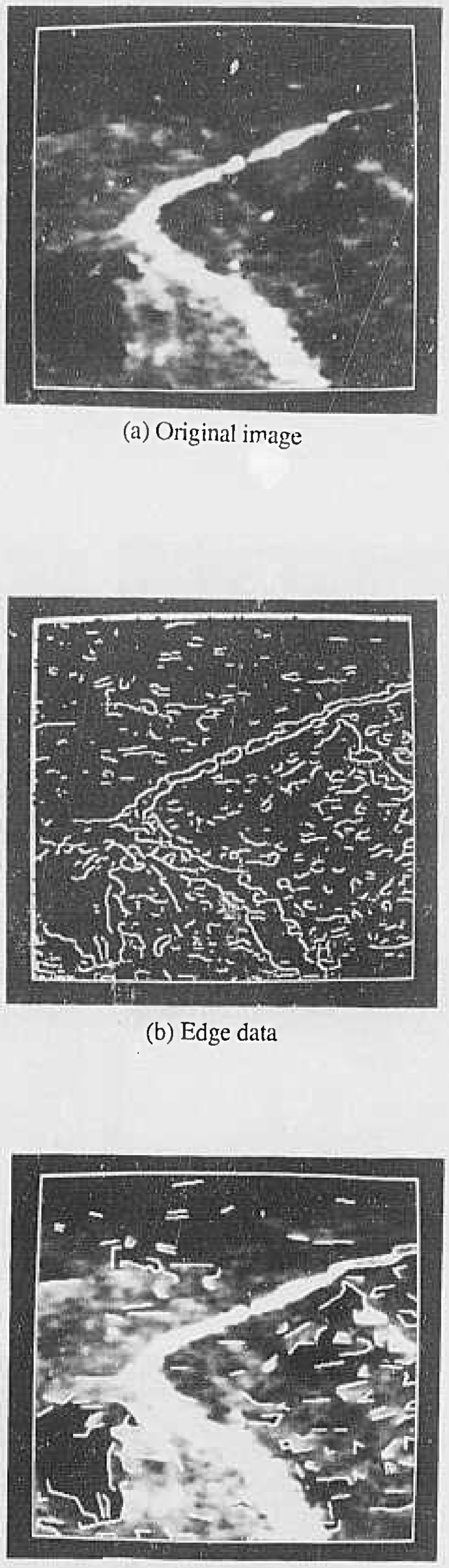

Initial hypotheses
Final classification
No new hypotheses
Fill slots with known objects
Generate new hypotheses to fill remaining slots
Hypotheses with sufficient evidence become established
Optimize established hypotheses' slots from possibilities
Subclassify established hypotheses to generate new hypotheses
Eliminate unlikely new hypotheses

Figure 4. Overview of the classification process.

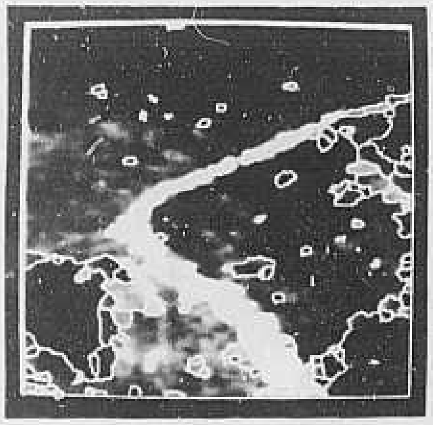


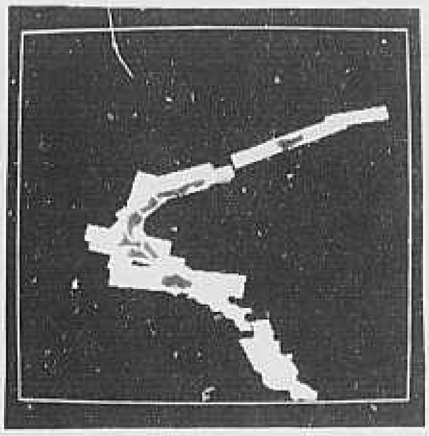
(a) Original image

(b) Edge data

(c) Extracted lines

Figure 5


(d) Segmentation


(c) Roads, road-subregions and line-roads

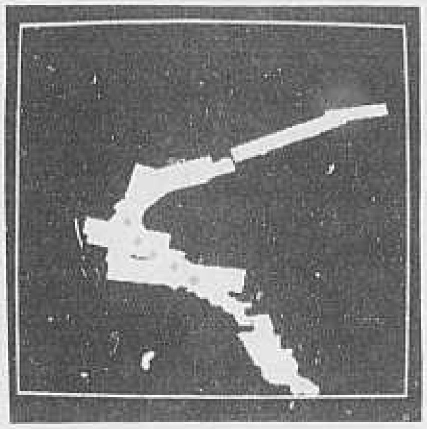
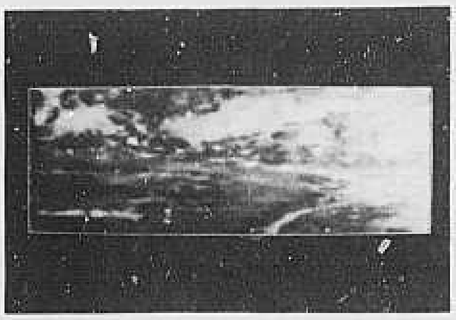
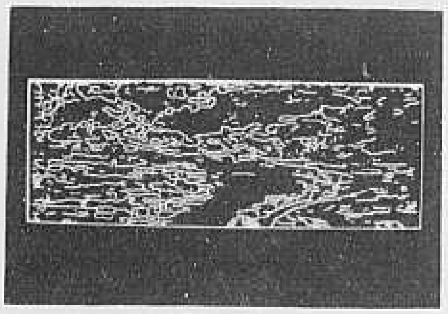
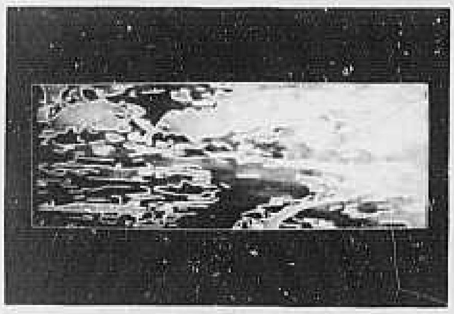

(f) Roads and vehicles

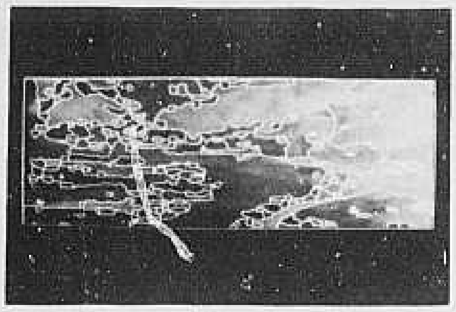
Figure 5, continued

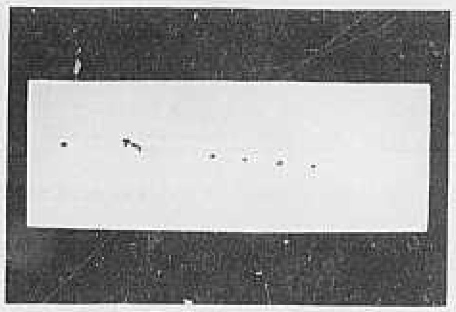
(a) First frame

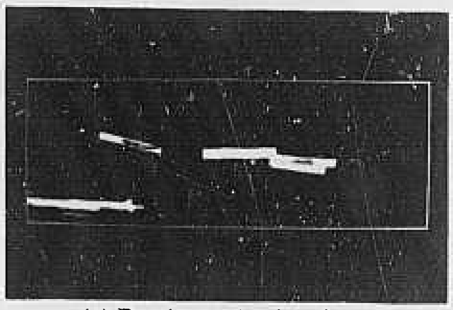

(b) Second frame


(c) Edge data


(d) Extracted lines


(e) Segmentation


(f) Difference image


(g) Roads, road-subregions and line-roads

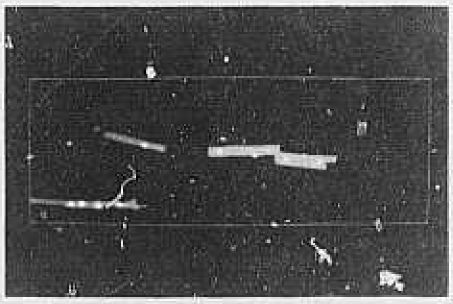

(h) Roads and vehicles

Figure 6

Precompiling a Geometrical Model into an Interpretation Tree for Object Recognition in Bin-picking Tasks

Katsushi Ikeuchi

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, PA 15213

Abstract

Given a 3D solid model of an object, we first generate apparent shapes of an object under various viewer directions. Those apparent shapes are then classified into groups (representative attitudes) based on dominant visible faces and other features. Based on the grouping, recognition algorithms are generated in the form of an interpretation tree. The interpretation tree consists of two parts: the first part for classifying a target region in an image into one of the shape groups, and the second part for determining the precise attitude of the object within that group. We have developed a set of rules to find out what appropriate features are to be used in what order to generate an efficient and reliable interpretation tree. Features used in the interpretation tree include inertia of a region, relationship to the neighboring regions, position and orientation of edges, and extended Gaussian images.

This method has been applied in a task for bin-picking objects which include both planar and cylindrical surfaces. As sensory data, we have used surface orientations from photometric stereo, depth from binocular stereo using oriented-region matching, and edges from an intensity image.

1. INTRODUCTION

Sensory capabilities will extend the functional range of robots. Without sensing the outer world, robots can only repeat pre-programmed tasks. Thus, the task is very rigid; such a system cannot overcome any small disturbance. Therefore, sensory capability is an essential component of a flexible robot.

Vision could be the most important type of robotic sensor. Since a vision sensor is a non-contact sensor, information can be obtained without disturbing the environment. Also, vision can acquire global information about a scene; this is not the case for a tactile sensor.

There are basically three tasks where the vision feedback can play an essential role:

1. finding the target object and determining the grasping points,
2. bringing the object from its initial point to a destination point while avoiding collision with other objects, and
3. assembling something using the object.

This research was sponsored by the Defense Advanced Research Projects Agency, DOD, through ARPA Order No. 4976, and monitored by the Air Force Avionics Laboratory under contract F33615-84-K-1520. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or of the U.S. Government.

This paper describes a method for visual guidance of a manipulator in the first task domain: finding an object. A manipulator without vision can only pick up an object whose position and attitude are pre-determined. Such a system needs the help of another machine or a human for feeding objects at a pre-determined place in a pre-determined attitude. Since this feeding job is tedious, it is quite unsuitable for a human being. Traditional mechanical feeding methods rely on a known part geometry to orient the part by forcing it through a sequence of gates, rails and stops. Besides being inflexible and capable of dealing with a very limited number of part types, these methods, including vibration, may cause defects in the objects due to collision.

Historically, bin-picking tasks have been attacked by detecting brightness changes [1-10]. Detecting brightness changes gives boundaries between regions corresponding to the objects. The boundaries obtained are compared with internal models to determine the attitude of the object. These edge-based approaches work particularly well with isolated objects lying on a uniform background provided the objects only rotate in the plane of support. In other words, these algorithms work well on binary images. However, such methods have difficulties extracting the contour of an 3D object from the image of a set of overlapping objects, which is typical in bin-picking.

Kelly and others [11] highlight scenes to segment and determine the position and the orientation of an object in a bin. This system is limited to cylindrical workpieces with a metallic surface. Also, their vision system only determines two degrees out of three degrees of freedom in attitude.

We [12-14] have presented techniques for using photometric stereo and an extended Gaussian image to determine object attitude. The photometric stereo determines surface orientations from the images under three different illumination conditions. A brightness triple at each point determines the surface orientation there. Distortions in brightness values due to mutual illumination or shadowing between neighboring objects are detected by the method as *un-interpretable* brightness triples. The locations of these triples are used to segment the visual scene into isolated regions corresponding to different objects. The distribution of surface orientations---an orientation histogram---measured over one of these isolated regions is used to identify the shape from a catalogue of known shapes. The object's attitude in space is also obtained as a by-product of the matching process. This system can pick up such a simple object as a doughnut successfully. This method, however, has three problems:

1. It is often difficult to express a complicated object such as a machine part with a mathematical function from which the extended Gaussian image is derived.
2. The extended Gaussian image is sometimes not powerful enough to determine the attitude of a

machine part due to self occlusion, narrowness of observable areas, or scatter of observable regions of the object due to self shadowing.

3. The previous system lacks a general representation of the outer world from which a planner can easily make a grasp plan. The purpose of robot vision is to provide the outer world information to task-achieving parts. The representation can serve as the starting point to the task-achieving module. Thus, the representation should be somehow a copy of the outer world and be in a convenient form to operate with it.

This paper resolves these problems using a geometrical modeler. This system has following characteristics:

1. An interpretation tree is precompiled from a object model so as to determine attitude by using the optimal features at each determining process.
2. The interpretation tree classifies a target region into a representative attitude, and then determines the attitude more precisely over the attitude range of the representative attitude.
3. The attitude and the position obtained is represented in the world in a geometrical modeler.

2. DERIVING THE INTERPRETATION TREE

A three-dimensional object varies its apparent shape depending on the viewer direction and rotation. All of the infinite 2D shapes of a 3D object are grouped into a finite number of topological equivalence classes. These equivalence classes are called *aspects* [21]. A shape change between two aspects is called an aspect change. Figure 1a shows an example of an aspect change. Between these two shapes, the two set of visible features, for example visible faces, are different. On the other hand, two different shapes within the same aspect are transformed with a linear transform. A shape change within the same aspect is called a linear change. Figure 1b shows an example of a linear change. Between these two shapes, the two sets of visible features are the same. Only the shape of each face is skewed.

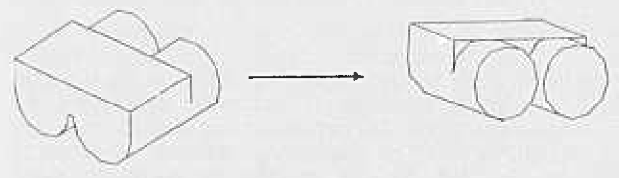


Figure 1a An example of an aspect change

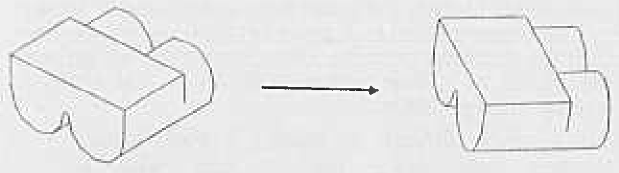


Figure 1b An example of a linear change

Figure 1 Two classes of shape changes

Different features are appropriate for resolving aspect changes and linear changes. Also different features are necessary for resolving linear changes depending on the aspect. Thus, it is desirable to precompile a geometrical model into an interpretation tree so that the most appropriate features among available features at each determination stage are used to resolve the aspect and linear changes.

Since there are two classes of shape changes, the interpretation tree also consists of two parts: resolving the aspect changes, and resolving the linear changes. In order to derive the interpretation tree, we will follow the next four steps: the first three steps for the aspect change and the last step for the linear change.

1. Extracting all possible aspects of an object from a geometrical model
2. Deriving classification branches of the aspect changes
3. Determining features to be used at the branching nodes of the tree
4. Determining features to be used to determine linear change

2.1. RESOLVING ASPECT CHANGE

2.1.1. Camera Model

The camera model is necessary to explore the shape change. The distance between the object and the camera is assumed to be far. This is true for most industrial vision applications because because the size of the object is small compared with the distance between the camera and the object. Under this assumption, the camera model is assumed to be orthographic projection. All lines of sight are parallel and perpendicular to the image plane.

The distance between the object and the camera is also assumed to be predetermined. This is also true for most industrial vision applications because the camera is usually fixed high above a pallet, a conveyer belt, or bin. Under this assumption, the object attitude and the viewer configuration have only three degrees of freedom. Two degrees of freedom exist in the viewer direction; the direction of the line of sight has two degrees of freedom with respect to the object. The remaining freedom exists in the rotation around the line of sight.

Note that we use the terms "object attitude" and "viewer configuration" interchangeably. "Object attitude" is the term with respect to the viewer centered coordinate system and "viewer configuration" is the term with respect to the object centered coordinate system. In our discussion, we only need a relative relationship between the viewer and the object. Thus, if we can specify the viewer configuration with respect to the object centered coordinate system, then we can also specify the object attitude with respect to the viewer centered coordinate system using the inverse transformation.

2.1.2. Extraction of Aspects

The aspects can be defined with various cues. Some researchers explore the aspects with visible lines [20-22]; others explore the aspects with visible vertices [23]. Occluding boundaries are also explored [24,25]. This paper explores the aspects using faces detectable by photometric stereo [41,35], because they are stable and contain rich geometrical properties. Photometric stereo can determine the surface orientation at the place where the three light sources project their light directly. This paper categorizes the aspects based on this observable faces by photometric stereo.

In the following discussion, we use the term "visible" for the sake of simplicity. In order to simplify the discussion, let us consider a convex object. The visibility of one face is determined by the relationship between the surface orientation of the face and the direction of the line of sight. When the angle between them is less than 90 degrees, then the surface is visible; otherwise, the surface is not visible. On the other hand, since the geometry between TV camera and the light sources is fixed in photometric stereo, the detectability of one face is also determined by the relationship between the surface orientation of the face and the line of sight of the TV camera. When the angle between the surface orientation and the line of sight is less than a certain limit angle, the surface is detectable; otherwise, the surface is not detectable [12]. Thus, we can assume that the detectability of one face is same as the visibility of the face, provided that detectable directions of one face become a cone of the lines of sight whose center is the surface orientation, while visible directions of one face become a hemisphere of the lines of sight.

The viewer configuration consists of three degrees of freedom; two degrees of freedom in viewer direction, and one degree of freedom in viewer rotation. Among the three degrees of freedom, some of the changes of viewer direction cause the aspect changes. On the other hand, changes of viewer rotation around the line of the sight do not cause aspect changes; they only cause linear changes. Thus, categorization of aspect requires only exploring apparent shapes under possible viewer directions. Viewer directions have two degrees of freedom and each can be described as a point of the Gaussian sphere at whose center a target object is located. Thus, classifying points on the Gaussian sphere gives possible aspects.

Each viewer direction and, thus, each point on the Gaussian sphere can be characterized by those faces visible from that direction. Let us suppose that

$$X_i = \begin{cases} 1 & \text{face i is visible} \\ 0 & \text{face i is not visible} \end{cases}$$

$(X_1, X_2, \ldots, X_n)$ denotes one label of an apparent shape based on the visible faces. Here, one face corresponds either to a planar surface, or a curved surface. (Moreprecisely, a face is defined as a region over which all derivatives are continuous; at the boundary of the face, some order of its derivatives become discontinuous.) We can characterize each viewer direction with this label. This label will be referred to as a *shape label*.

The set of viewer directions that have the same *shape label* becomes an *aspect*. There are two ways to generate possible aspects: an analytic method, and an exhaustive method. If the target object is a convex polyhedron, then the analytic method is easy. The face visibility is determined by the relationship between the viewer direction and the surface orientation of the face. Each viewer direction can be described as a point; in the meantime, each surface orientation can also be represented as a point on the same Gaussian sphere. Then, the visible viewer directions of a face are limited by a circle on the Gaussian sphere. The circle center corresponds to the surface orientation of the face, and the radius of the circle is $\pi/2$. The inside area of the circle corresponds to the viewer directions which can observe the face with the photometric stereo. Drawing these visibility circles on the Gaussian sphere, the combination of the circle covers on the sphere gives the possible face labels and, thus, possible aspects.

If the target object is non-convex, then the visibility circle is distorted due to self occlusion and the analytic method becomes difficult. Curved surfaces are also difficult for the analytic method. Thus, in the general case, the exhaustive method is preferable. Essentially, the exhaustive method generates various apparent shapes of the object under various viewer directions, and then examines shape labels of the generated shapes in order to classify them into aspects.

The Gaussian sphere to express viewer directions is tessellated evenly for efficient sampling; a geodesic dome is used to tessellate the Gaussian sphere evenly into many small spherical triangles [26]. Each tessellated triangle corresponds to a particular viewer direction. These sampled viewer directions evenly cover the whole Gaussian sphere surface.

At each sampled viewer direction, an apparent shape of the object is generated using a geometrical modeler. Then, we can sample all possible apparent shapes evenly under all possible viewer directions over all small triangles of the geodesic dome. One observable shape gives one shape label, $X_1, X_2, \ldots, X_n$. After obtaining all shape labels of all generated shapes, aspects are generated based on these shape labels so that shapes at each aspect share the same shape label.

One representative attitude will be selected from each aspect and each aspect is represented by its representative attitude. That is, the viewer directions over one particular range are represented by one representative attitude. Usually, the viewer direction which gives the largest sectional area within the aspect is selected as the viewer direction for the representative attitude. The viewer rotation for the representative attitude is determined so that the maximum inertia direction agrees with the x axis on the image plane.

Figure 2 shows an example of this process. Figure 2a is a picture of an object. Figure 2b is a model synthesized using a geometrical modeler. The Gaussian sphere to represent the possible viewer directions is tessellated into small triangles using the one-frequency dodecahedron shown in Figure 2c. Apparent shapes are generated at the viewer directions corresponding to the centers of the triangles. Since the one-frequency dodecahedral geodesic dome has sixty triangles, sixty different shapes are generated as shown in Figure 2d, where the faces enclosed with bold lines are detectable by the photometric stereo. These observable faces are the faces where the three light source project light directly. Note that even though some faces are visible to humans, they cannot be detected by the photometric stereo because of the geometry of the light sources. Such faces are enclosed with thin lines. Figure 2e shows the larger eight faces used for the shape label among the twelve component faces of the object. Note face 1 and face 2 in Figure 2e are treated as one continuous surface even though they are divided into small patches approximating cylindrical surfaces. The shapes in Figure 2e are combined into seven aspects as shown in Figure 2f. Smaller regions under a certain threshold are regarded as non-detectable. The numbers of the visible faces as well as the shape labels are printed under the aspect number in Figure 2f. For example, in aspect 1, face 1, face 2, and face 3 are observable. Thus, the shape label of aspect 1 becomes 11100000. For aspect 1 to aspect 5, five representative attitudes are generated as shown in Figure 2g. Aspect 6 corresponds to a hole region of the object and such a steep convex area cannot be detected by photometric stereo. Aspect 7 has too small a visible area. Thus, no representative attitudes are generated from the aspects 6 and 7.

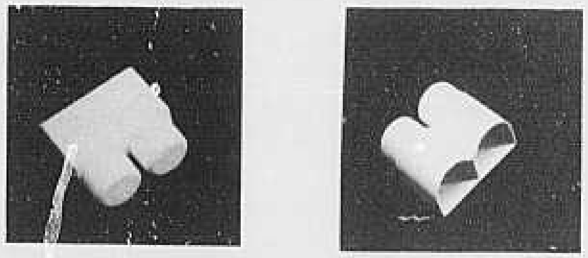


Figure 2a An object.

Figure 2 An object and its representative attitudes.

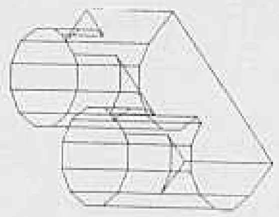


Figure 2b A synthesized model of the object.

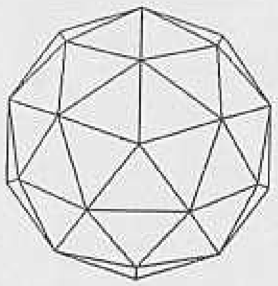


Figure 2c One-frequency dodecahedron.

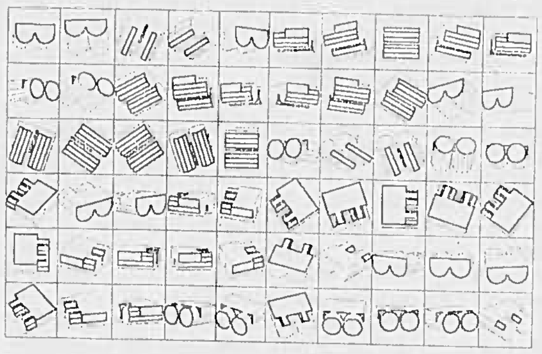


Figure 2d Sixty apparent shapes of the object.

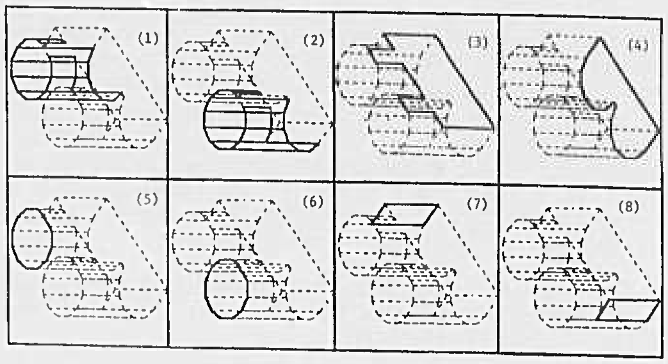


Figure 2e Eight identifying faces.

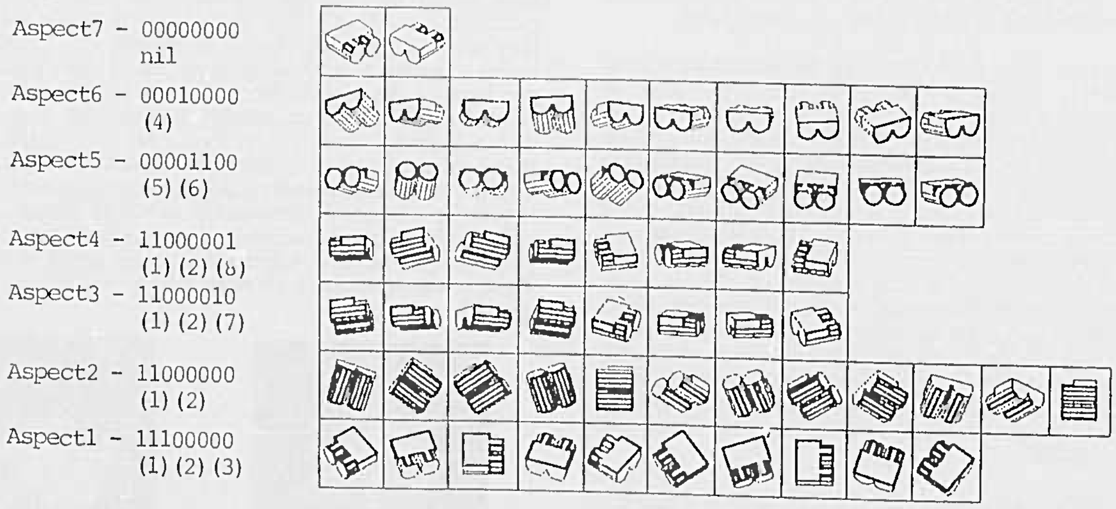


Figure 2f Seven aspects.

Figure 2 (continued)

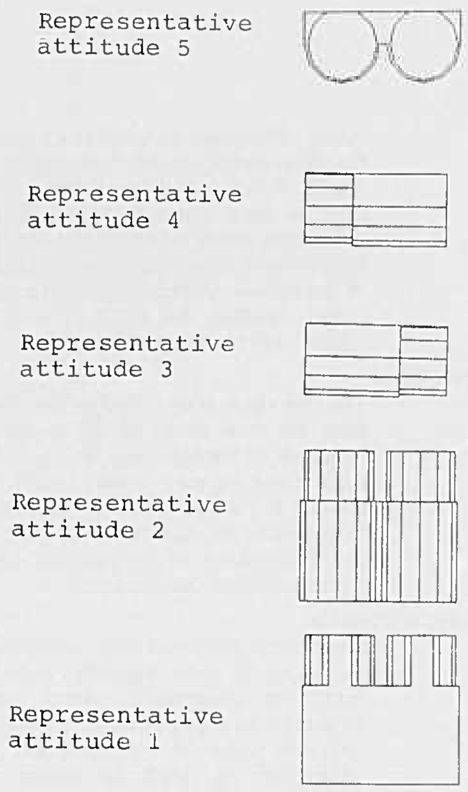

Representative attitude 5
Representative attitude 4
Representative attitude 3
Representative attitude 2
Representative attitude 1

Figure 2g Five representative attitudes.

Figure 2 (continued)

2.1.3. Branching into the Final Aspect

The previous section gives the final stage in shape classification: the deepest leaf of the interpretation tree. Yet we have not determined the branches of the interpretation tree from the root to the leaves. Therefore, the next step is to generate branches from the root to the final aspects. The leaves of the tree correspond to the final aspects, while the root corresponds to the unclassified stage.

Branches are also generated using the shape label. The shape label and the aspects depend on the faces which generate the shape labels. By using these characteristics, the branches will be generated. At first we will put the faces of an object in area order; $f_1, f_2, \ldots, f_n$. Then, we will consider the subsets of the face groups $g_1=\{f_1\}, g_2=\{f_1 f_2\}, \ldots, g_n=\{f_1 f_2 \ldots f_n\}$. One set of shape labels is obtained from $g_1=\{f_1\}$, and one set of intermediate aspects is obtained. In the same way, another set of shape labels is obtained from $g_2=\{f_1, f_2\}$ and, thus another set of intermediate aspects is obtained. Following this method, n different sets of intermediate aspects are obtained from the sequence of face groups, $g_1, g_2, \ldots, g_n$. The sequence of intermediate aspects given by this sequence of face groups generates a tree whose leaves are the final aspects.

Note that only valid intermediate aspects are generated among the possible combinations of shape label at each face group. These valid intermediate aspects and valid shape labels are obtained from a geometrical modeler using the same method as for generating the final aspects described in the previous section. If we follow a brute force method to generate a tree whose branches correspond to conditions whether faces of the object are visible or not without considering the validity of each shape label, the method generates a tree of 2^n leaves. On the other hand, since our method generates only valid shape labels of each face group based on the object, the method generates only leaves corresponding to the valid final aspects.

The fact that the sequence of valid intermediate aspects generates a tree that contains only the final aspects at the leaves can be proved inductively:

Proof:

(Step 0) g_1 is a subset which consists of only one face f_1, which is the largest among the faces of the object. This subset generates a shape label x_1. Using this label the Gaussian sphere is divided into two parts. Under any viewer direction in the part which has $x_1=1$, the photometric stereo can observe the largest face 1; under any viewer direction in the part which has $x_1=0$, the photometric stereo cannot observe the largest face 1. Thus, the first step will generate two intermediate aspects from g_1. Note that if either part of the two part has no valid intermediate aspect, the branch will not be generated.

(Step n) Next we will consider the relationship between the intermediate valid aspect from g_i and the intermediate valid aspects from g_{i-1}. The intermediate valid aspect of g_i is obtained by dividing the intermediate valid aspect of g_{i-1} based on the visibility of the face, f_i. Thus, if the number of aspects increases from $i-1$ to i, new aspects at i come from only division of aspects at i; no new aspects at i come from combining one part of one aspect at $i-1$ and one part of the other aspect at $i-1$.

This division sequence generates a tree structure which gradually reaches the final aspects. Since the representative attitudes are generated using the shape label of g_n, there is always one and only one leaf corresponding to each representative attitude in the final tree. This tree structure will be used as the structure of the interpretation tree.

Figure 3 shows the branches obtained from the object shown in Figure 2. In the application, it often occurs that two faces have the same area. Since our method of tree generation is based on the area size of each face, the method becomes unstable at that branch. In this case, at the first step, we will divide the intermediate aspect into aspects; any one of the faces are observable (xx), and none of the faces are observable (00). Then, (xx) aspect is divided on the visibility of the faces. This is because we will divide the resembling aspects at a later stage. The B0 branch corresponds to the two cylindrical surfaces, B1 corresponds to the wide planar surface, B2 corresponds to the hole region, B3 corresponds to the two circular surfaces, and B4 and B5 correspond to the side planar surfaces. These branches divide the Gaussian sphere into seven final aspects.

2.1.4. Work models

The work models consist of physical face information. Work models will be used to classify one target region into an aspect, and to determine the attitude of an object observed as the target region. These work models are derived from a geometrical modeler in the modeling process, and are derived from needle maps and/or edge maps in the determining process.

The work models are generated at each representative attitude. Since the surface orientation is available at each region from the needle map, the original face information can be recovered from the observed region information using an

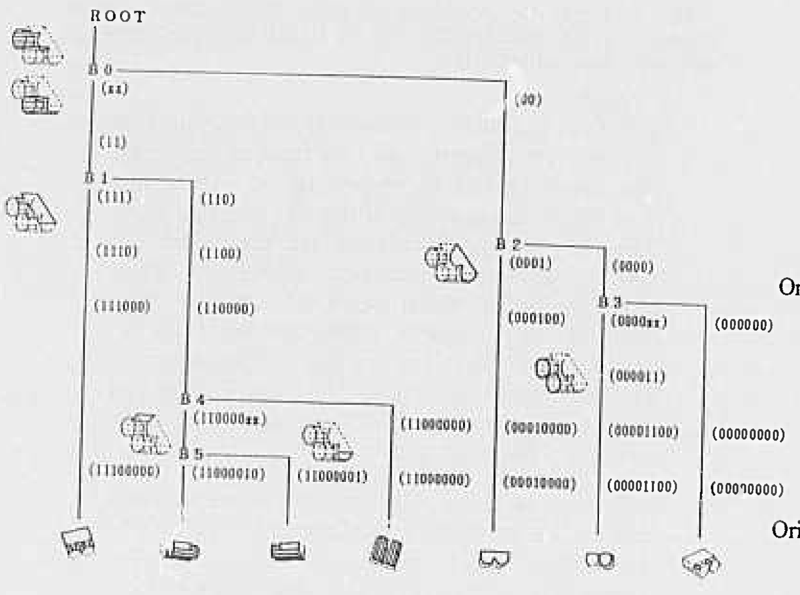


Figure 3 Branches based on the shape label.

affine transform, where we assume orthographic projection as the camera model. For example, when the surface orientation, the affine matrix, and the observed region shape are known, the original face shapes can be recovered from the skewed region shape with the affine transform. Information for only one attitude is necessary at each aspect in which detectable faces are the same and they are reachable from each other by the affine transformation. The work models are thus generated at each representative attitude which represents one aspect.

Let (p,q) be the surface orientation of one face.

$$T=\begin{bmatrix} 1+p^2 & (pq)/(1+p^2) \\ 0 & (1+p^2+q^2)/(1+p^2) \end{bmatrix}$$

gives the affine matrix to recover the original face information from the observed face information. Thus, given (p,q) from photometric stereo, we can derive T and transform apparent features to original features.

Our work models consist of original face inertia, original face relationship, original face shape, original edge relationship, extended Gaussian image, and surface characteristic distribution.

Original face inertia

The inertia moments of one face in the directions of least and most inertia direction. These inertia moments give the rough shape information of the face. See Appendix I for more details.

Original face relationship

A non-convex object often appears as multiple isolated regions under the photometric stereo. In this case, the relationships between regions are used as a work model.

For each region, the relative position of other regions are stored. The relative position is described by a vector whose length corresponds to the distance between the mass centers of the two regions and whose direction indicates the direction from the mass center of the region to the other mass center based on the maximum inertia direction and the surface orientation of the region. If the region has no unique inertia direction, for example, a circular region, only the distance is stored.

Original face shape

The face shape is described as the distance from the mass center of the face to the boundary of the face as a function of the angle round the mass center, $d=d(\theta)$. The rotation angle θ is calculated with respect to the maximum inertia direction. This is a two dimensional well-tessellated surface representation of the shape [26].

Original edge relationship

Some of the prominent edge information is also used. In some cases the needle map from the photometric stereo cannot determine the object attitude uniquely. In this case some of the prominent edge information is used to reduce this ambiguity. Thus, some of the edge information is stored if necessary.

The edge information is described by the starting position and the ending position.

These positions are denoted relative to the mass center of the face and the maximum inertia direction. To apply this information, a position is converted into the position on the image plane using the affine matrix. Then, the area connecting the converted starting position and the converted ending position will be searched on the edge map to determine whether there is an edge or not.

Extended Gaussian image

Roughly speaking, the extended Gaussian image of an object is a spatial histogram of its surface orientation distribution [28-32]. Let us assume that there is a fixed number of surface patches per unit surface area, and that a unit normal is elected on each patch. These normals can be moved so that their "tails" are at a common point and their "heads" lie on the surface of a unit sphere. This mapping is called the Gauss map; the unit sphere is called the Gaussian sphere. If we attach a unit mass to each end point, we will observe a distribution of mass over the Gaussian sphere. The resulting distribution of mass is called the extended Gaussian image (EGI) of the object.

The EGI has the following properties:

1. Neither the surface normal nor the Gauss map depend on the position of the origin. Thus, the resulting EGI is not

affected by translation of the object.

2. When an object rotates, its EGI also rotates. However, the EGI rotates in the same manner as the object. In other words, this rotation does not affect the relative EGI mass distribution over the sphere.

Surface characteristic distribution

The surface characteristic distribution is available from the surface orientation distribution. A surface patch has a characteristic such as planar, cylindrical, elliptic, or hyperbolic. The first and the second fundamental forms can be obtained from the surface orientation and its derivatives, and from these the Gaussian curvature and the mean curvature are obtained [33,34]. The characteristic, defined in terms of the Gaussian curvature and the mean curvature are independent of the viewer direction and the rotation. Distribution of the characteristics are used as work models. See Appendix I for more details.

2.1.5. Classification rules

This section gives rules to generate the classification part of the interpretation tree. At each branch we examine whether one of the rules can discriminate between two intermediate aspects. If one of the rules can discriminate, the rule is registered at that branch. The decision whether or not the rule can divide them is made by a human at present.

L1: *Comparison based on the original face inertia.*

L2: *Comparison based on the original face shape.*

L3: *Comparison based on the extended Gaussian image.*

L4: *Comparison based on the surface characteristic distribution.*

L5: *Comparison based on the edge distribution.*

L6: *Comparison based on the region distribution.*

L7: *Comparison based on the relationship between a particular edge and a particular surface characteristic distribution.*

If the observed shape of an object cannot be classified into an aspect with these rules, it means that the object is observed with the same number of regions whose area sizes, inertia moments, edge distributions, and surface characteristic distributions are identical in two different aspects. Such objects are beyond the scope of our technique.

Deriving the classification part of the tree

The classification part of the interpretation tree, Figure 4, is generated for the object shown in Figure 2a. At the B0 branch, the rule L1 (original face inertia) can divide all the attitude groups into two attitude groups. At the B1 branch, rule L1 can divide the attitude groups. Both B2 and B3 have branches at which the attitude groups are not visible. Thus, these branches are pruned.

At branch B4, none of L1 (inertia), L2 (shape), L3 (EGI), L4 (characteristic), L5 (edge) can divide the attitude groups. L6 (topology) can divide the branch. L7 (edge-region) can discriminate the attitude groups at the branch.

Thus, B0-L1, B1-L1, B2-pruned, B3-pruned, B4-L6, B5-L7 are adopted into the interpretation tree. Since B0 and B1 branches have the same rule and they are consecutive, they are joined into a three-branch node.

2.2. RESOLVING LINEAR SHAPE CHANGE

2.2.1. Determination rules

This section gives the rules to generate the part of the interpretation tree which determines the viewer direction and the rotation. Each rule that can reduce some of the remaining freedom in the viewer direction and rotation will be adopted into the tree. The decision whether or not the rule can reduce the freedom is made by a human at present.

A1: *Using the mass center of EGI mass distribution.*

A2: *Using the extended Gaussian image.*

A3: *Using the position of observable areas distribution.*

A4: *Using the inertia direction of original face.*

A5: *Using the rotation of original face shape.*

A6: *Using the position of the surface characteristics distribution.*

A7: *Using the position of the edges.*

A8: *Using the position of the edges with respect to the position of the surface characteristics distribution.*

If we cannot determine the viewer direction and the rotation with these rules, it means that the object is observed with the same number of regions whose area sizes, inertia moments, edge distributions, and the surface characteristic distributions are identical in two different attitudes. Such objects are beyond the scope of our technique.

The viewer direction and rotation are determined at aspect using the most effective feature at each step. The most powerful rule for determining the viewer direction and rotation depends on the aspect and the stage of the determining process. Thus, we will discuss which rule will be used for generating the determination part of the interpretation tree at each aspect.

Aspect S1

The main visible part of this aspect is a planar surface. A1 (EGI mass center) can determine the viewer direction, while viewer rotation can be constrained with neither A1 nor A2. More precisely, since the observable region of aspect S1 is a planar surface, both the EGI and the EGI mass center position [29] can determine the viewer direction uniquely. However, neither the EGI distribution nor the EGI mass center over the planar surface can constrain the viewer rotation around the viewer direction. Thus, the other rules should be applied to determine the viewer rotation.

Since the aspect has only one observable region, A3 (region distribution) cannot be applied to this S1 aspect. A4 (inertia direction) can constrain the viewer rotation up to two directions. Between the two directions, A5 (original face shape) can determine the viewer rotation uniquely. Thus, A1 (EGI mass center), A4 (inertia direction), and A5 (original face shape) are adopted into the tree to determine the viewer direction and the rotation at the aspect, S1.

Aspect S2

This aspect has two observable regions of cylindrical surfaces. A1 (EGI mass center) can determine viewer direction, while the viewer rotation cannot be constrained with A1.

Theoretically, the EGI distribution can determine the viewer direction and the rotation uniquely in this representative attitude. However, the determined rotation is very noisy. Thus, we will use the other features to determine the viewer rotation.

Since this aspect has two observable regions, A3 (region distribution) is applicable and can constrain the viewer rotation up to two directions. None of A4 (inertia direction), A5 (original face shape), nor A6 (surface characteristic) can constrain the remaining freedom of the viewer rotation. A7 (edge distribution) can determine the viewer rotation uniquely. Thus, A1 (EGI mass center), A3 (region distribution), and A7 (edge distribution) are adopted into the tree.

Aspect S3

S3 aspect has one observable region which mainly consists of three parts: a planar surface patch and two cylindrical surface patches. A1 (EGI mass center) can determine the viewer direction, while the viewer rotation is difficult to determine in practice due to the same reason as with the aspect S2.

A3 (region distribution) cannot be applied to this aspect due to the single observable region. A4 (inertia direction) can constrain the viewer rotation up to two directions. Neither A5 (original face shape) nor A6 (surface characteristic) can constrain the remaining freedom. A7 (edge distribution) can determine the viewer rotation uniquely. Thus, A1 (EGI mass center), A4 (inertia direction) and A7 (edge distribution) are adopted into the tree.

Aspect S4

The features used to determine the viewer direction and the rotation are the same as those of the aspect A3.

Aspect S5

Aspect S5 has two regions observed separately which come from two planar surfaces. Thus, A1 (EGI mass center) can determine viewer direction, while the viewer rotation is difficult to constrain with A1 for the same reason as with aspect S1. Since this aspect has two observable regions, A3 (region distribution) is applicable and can constrain the viewer rotation up to two directions. None of A4 (inertia direction), A5 (original face shape), nor A6 (surface characteristic) can constrain the remaining freedom of the viewer rotation. A7 (edge distribution) can determine the viewer rotation uniquely. Thus, A1 (extended Gaussian image), A3 (region distribution), and A7 (edge distribution) are adopted into the tree. Figure 4 shows the interpretation tree obtained.

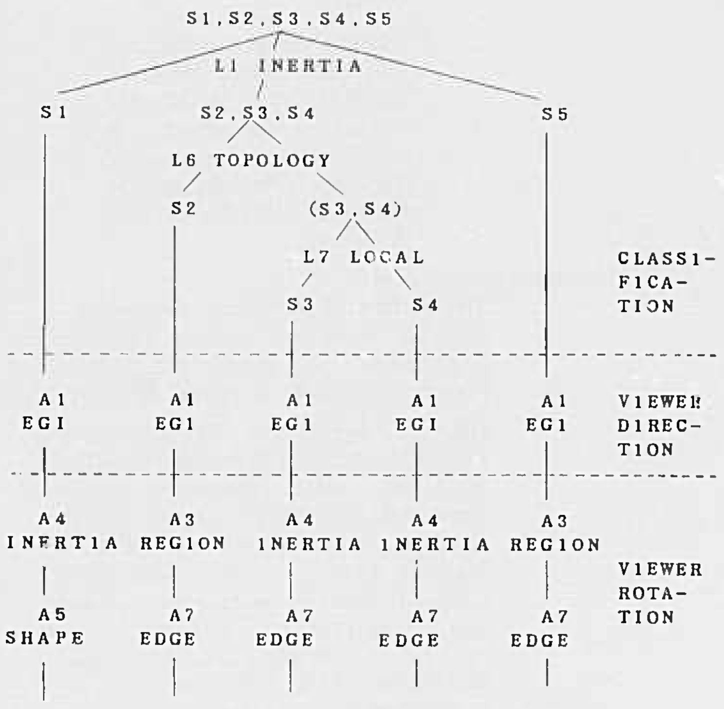


Figure 4 The interpretation tree.

3. APPLYING THE INTERPRETATION TREE

3.1. Attitude Determination by the Interpretation Tree

The system can use three kinds of maps: edge maps, needle maps, and one depth map. Three edge maps can be obtained by differentiating the three intensity maps also to be used for the photometric stereo. A needle map can be obtained by the photometric stereo system. A depth map can be obtained by comparing a pair of needle maps which are generated by a dual photometric stereo system [35]. The edge maps, the needle map, and the depth map are represented in the same coordinate system; that is, all pixels having the same X-Y coordinates correspond to the same physical point.

The highest region is determined from the depth map. This highest region will be sent to the interpretation tree as the target region. The interpretation tree extracts necessary features from the region. These features will be transformed according to the procedures defined in the interpretation tree. These transformed features will be compared with features in the work models defined in the interpretation tree. Following this procedure, the target region will be classified into one of the aspects, and then the precise attitude and position determined.

3.2. Case 1: Aspect S1

Figure 5 shows one of the input scenes, where the white arrow indicates the highest region. From this scene, the edge map shown in Figure 5b is obtained. The photometric stereo system gives the needle map shown in Figure 5c. Further, the depth map shown in Figure 5d is obtained by the dual photometric stereo system.

This highest region will be given to the interpretation tree. The interpretation tree calculates the inertia moment of the original face observed as the region (L1). The mass center and the region distribution can be obtained over the binary map which has been converted from the needle map to have 1 at the places where the surface orientation is determined, and to have 0 at the places where the surface orientation is not determined. Then, the affine matrix is obtained from the the surface orientation distribution over the region. Finally, the interpretation tree can determine the inertia moment of the original face using the affine matrix and the region distribution. Figure 6a shows the region distribution and the square which is displayed by the interpretation tree. The square has the same inertia and direction as the original face. The interpretation tree determines that this region belongs to the aspect S1 based on the inertia value.

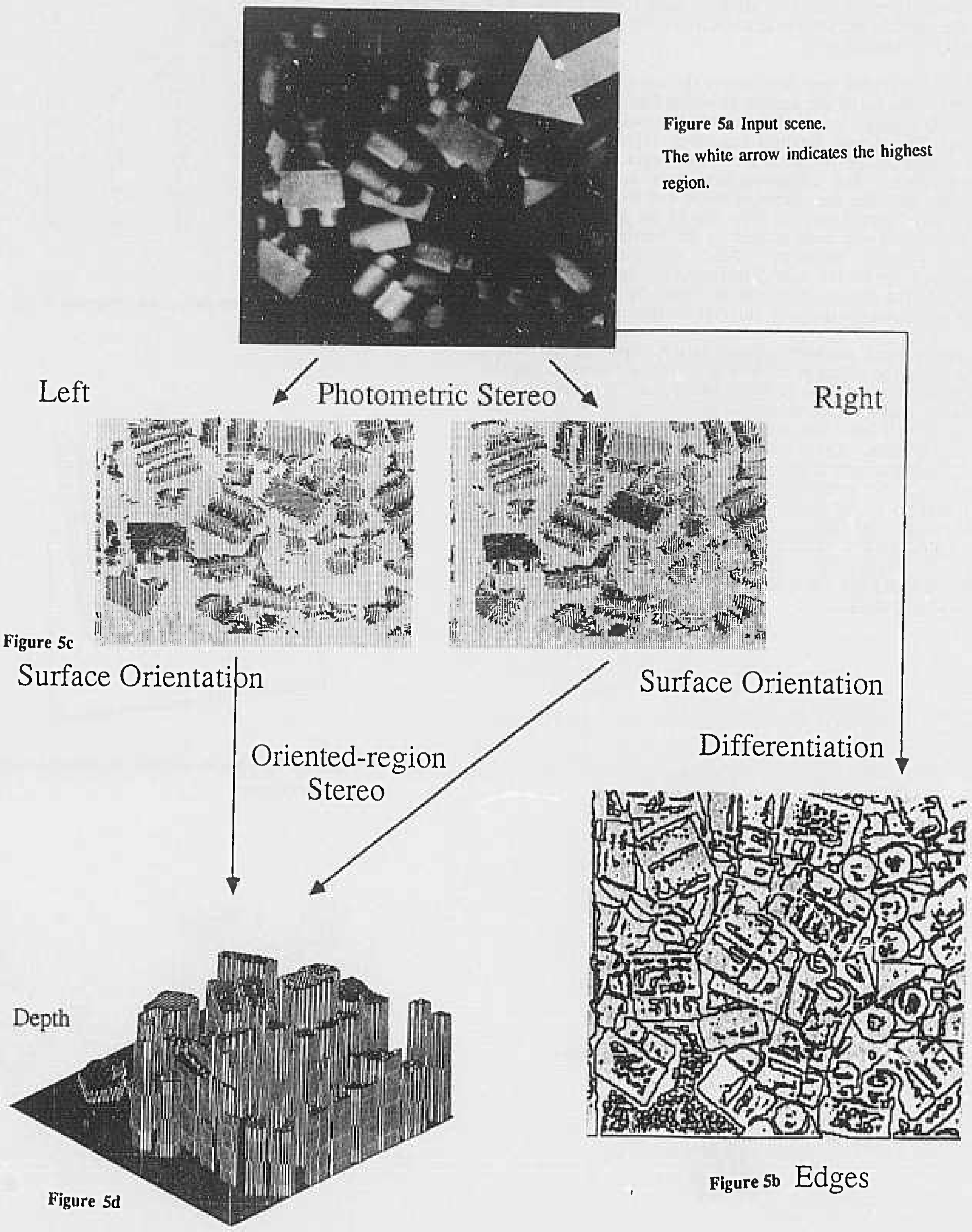


Figure 5a Input scene.
The white arrow indicates the highest region.

Figure 5c

Figure 5b

Figure 5d

Figure 5 Input scene and maps.

The interpretation tree uses the EGI mass center to determine the viewer direction (A1). This EGI mass center is obtained from the surface orientation distribution over the target region by the interpretation tree.

The interpretation tree determines the viewer rotation up to two directions using the inertia direction (A4). Branch A5 in the interpretation tree requires the original face shape to determine the viewer rotation uniquely. Figure 6b shows the original face shape obtained from the target region. In this case, however, the interpretation does not measure the difference between the observed shape and the shape from the models in all directions, but only checks the crack direction of the observed region with respect to the inertia direction under the two possible rotations. Since the viewer rotation is constrained up to the two directions, the interpretation tree determines the object attitude in the space by this comparison. Figure 6c shows the decision flow on the interpretation tree.

A geometrical modeler represents the object in the world model using the object position and the attitude obtained by the interpretation tree. The object position can be obtained from the depth map. Around the target region, there are a few regions which have not been processed by the interpretation tree at this time. These neighboring regions are expressed as dodecahedral prisms in the world model. The height of a prism agrees with the height of the corresponding region, and the cross section of the prism is an approximation of the region shape by the dodecagon. These dodecahedral prisms are also represented in the world model in a geometrical modeler as shown in Figure 6d. If we repeat the recognition-and-representation loop, the system finally obtain the representation shown in Figure 6e.

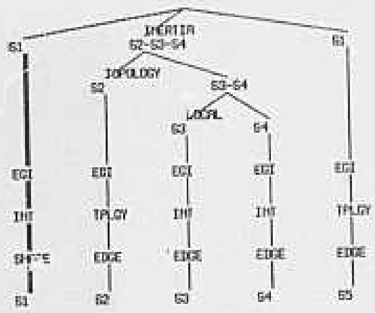


Figure 6c The decision path in the interpretation tree.

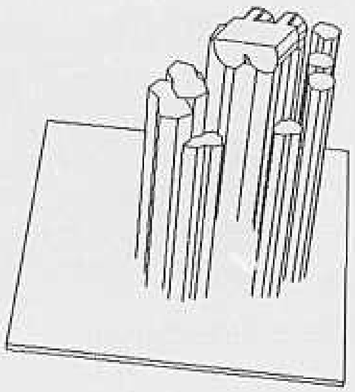


Figure 6d The obtained position, the obtained attitude, and the neighboring regions.

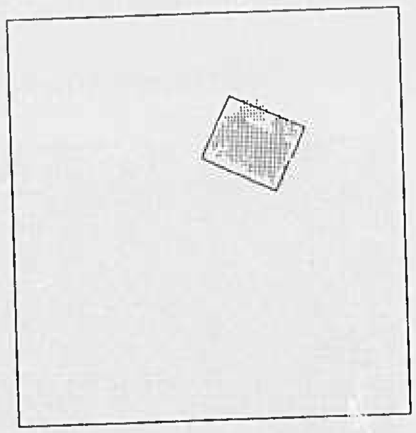


Figure 6a The target region and the original face inertia.

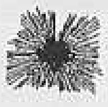


Figure 6b The original face shape recovered by the Affine transformation. The shape is represented using 2D WTS.

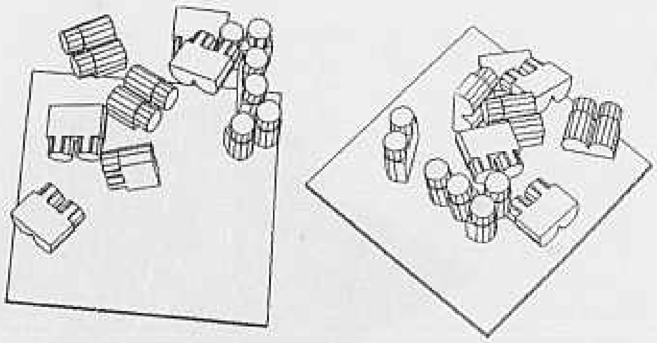


Figure 6e Scene description.

Figure 6 An interpretation example: aspect S1.

3.2.1. Case 2: Aspect S2

Figure 7a shows a second example. The white arrow in the picture indicates the highest region. The interpretation tree calculates the original face inertia of the region from the binary map converted from the needle map and the affine matrix obtained from the needle map over the target region. Figure 7b shows the square which has the same inertia direction and inertia value as the obtained inertia moment. The interpretation tree determines this region to belong to the group of aspects S2, S3, and S4 from the inertia value (L1).

The interpretation tree makes the distinction between the aspect S2 and the group of the aspects S3 and S4 by determining whether a brother region exists having the same inertia direction and the inertia value around the target region. The interpretation tree tries to find such a brother region; it succeeds, as shown in Figure 7b, where the target region and the brother region are connected with a solid line. From this evidence, the interpretation tree determines that the target region and the brother region come from the same object and belong to the aspect S2 (L6).

The interpretation tree makes an EGI-mass center comparison to determine the viewer direction (A1). From the direction of the brother region, the viewer rotation is determined up to the two directions (A3).

The edge distribution is necessary to determine the viewer rotation uniquely (A7). The interpretation tree only examines the existence of the edge distribution whose direction agrees with the edge direction under one of the two possible rotations, at the place where one of the two rotations is supposed to make the edge distribution. This predicted place and the predicted direction can be obtained by applying the affine transform to the edge representation in the work models. In Figure 7c, the dotted lines indicate the distribution of edges over the target region and the broken lines indicate the search areas for the edge distributions. The solid lines in Figure 6c indicate the edges found to have the supposed directions at the supposed places under two possible rotations of the object. One of the two rotations is determined by the comparison of the edge distributions. The interpretation tree determines the object attitude in the space uniquely up to this point. The decision flow on the interpretation tree is expressed as the bold line in Figure 7d. Figure 7e shows the object attitude obtained by this process.

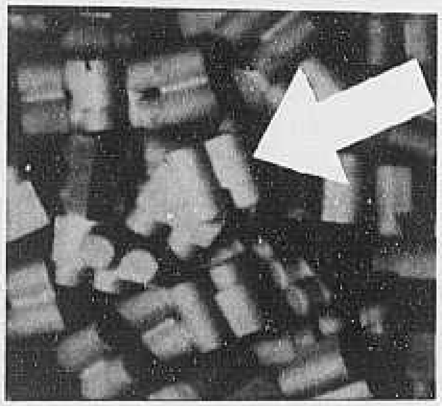


Figure 7a Input scene. The white arrow indicates the highest region.

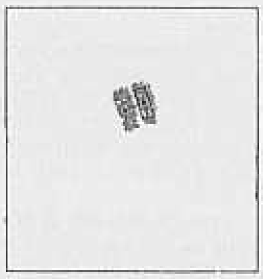
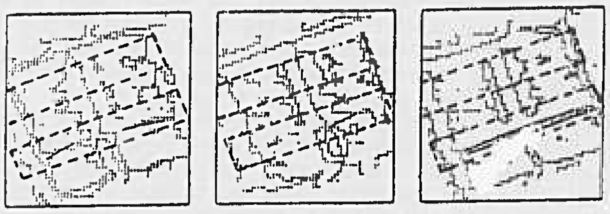


Figure 7b The target region and its brother region found by the algorithm.


Figere 7c Obtained edges. The interpretation tree only examines the existence of the edge distribution whose direction agrees with the edge direction under one of the two possible rotations, at the place where one of the two rotations is supposed to make the edge distribution. The dotted lines indicate the distribution of edges over the target region and the broken lines indicate the search areas for the edge distributions. The solid lines indicate the edges found to have the supposed directions at the supposed places under two possible rotations of the object.

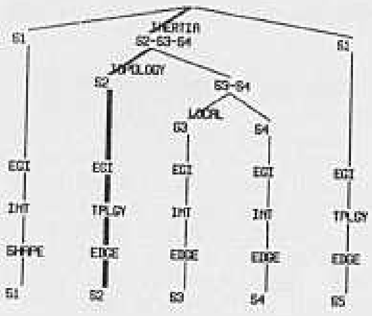


Figure 7d The decision flow on the interpretation tree.

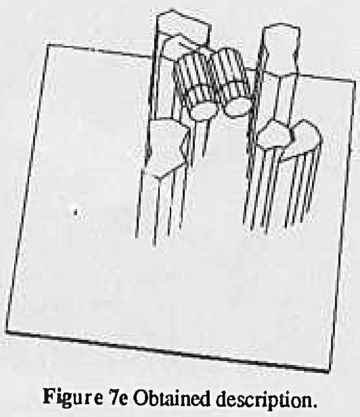


Figure 7e Obtained description.

Figure 7 An interpretation example: aspect S2.

3.2.2. Case 3: Aspect S4

Figure 8a shows the third example classified into the aspect S4. The white arrow indicates the highest region. The interpretation tree determines that the target region belongs to the group of the aspects S2, S3, and S4 based on the original face inertia. Figure 8b shows the target region and the obtained moment-compatible square of the original face.

The interpretation tree makes the distinction between the aspect S2 and the group of the aspects S3 and S4 based on the existence of a brother region (L6). Since there are no brother regions around this target region, the region is determined to belong to the group of the aspects S3 and S4.

The surface characteristic distribution with respect to the edge distribution resolves the ambiguity between S3 and S4 (L7). The interpretation tree examines which attitude has the more consistent surface characteristic distribution. First, the interpretation tree searches the existence of the edge distribution at the supposed places at the supposed directions from the inertia direction as in the S2 case. Figure 8c indicates the edge distribution found as the solid lines. Second, the interpretation tree generates both the surface characteristic distribution of S3 and that of S4 based on the inertia direction and the edge distribution.

The aspect S3 has the planar surface at the left region and the cylindrical surface at the right region with respect to the edge distribution shown in Figure 8c. Figure 8d shows the surface characteristic distribution which agrees with the distribution of the aspect S3. Note that since no distributions agree with the observed distributions, the result figure shows white space. On the other hand, if the target region is assumed to belong to the aspect S4, the region should have the cylindrical surface at the left region and the planar surface at the right region relative to the edge distribution. Figure 8e shows the characteristic distribution which agrees with the aspect S4. The interpretation tree determines that the target region belongs to the S4 aspect.

The interpretation tree determines the viewer direction from the EGI mass center (A1). The viewer rotation is determined up to the two directions from the inertia direction (A4). To determine the viewer rotation uniquely, the edge distribution is necessary (A7); it had been obtained when the system used rule L7. The interpretation tree determines the object attitude from these comparisons, while the object position is obtained from the depth map. Figure 8f shows the decision flow on the interpretation tree. Using the object position and attitude, the object is represented in the world model in a geometrical modeler shown in Figure 8g.

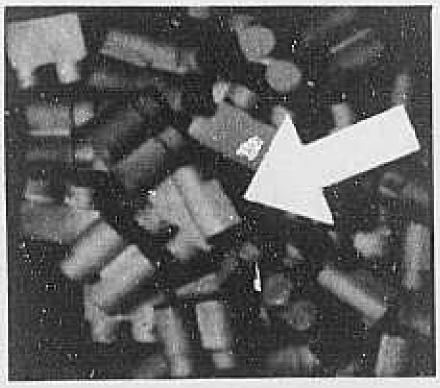


Figure 8a Input scene. The white arrow indicates the highest region.

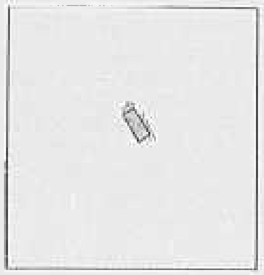


Figure 8b The target region and its original face inertia.

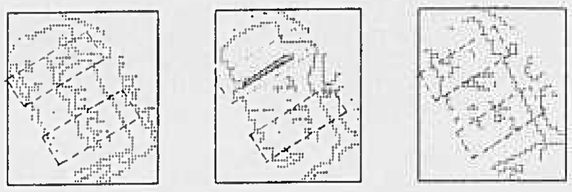


Figure 8c The edge distributions. The dotted lines indicate output from an edge operator. The broken lines indicate search areas predicted from the model. The solid lines indicates the edges which corresponds to the model.

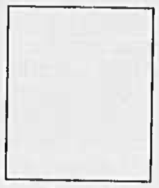


Figure 8d No surface characteristic distributions agree with the distributions of aspect S3.

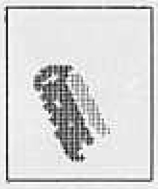


Figure 8e The characteristic distributions which agrees with aspect S4. The target region has the cylindrical surface at the left region and the planar surface at the right region relative to the edge distribution. This distributions correspond to aspect S4.

Figure 8 An interpretation example: aspect S4.

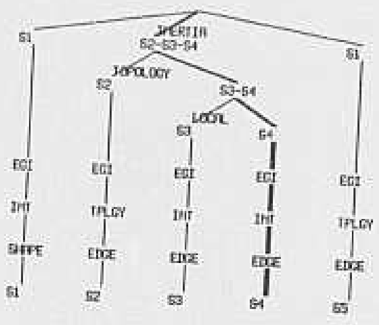


Figure 8f The decision flow on the interpretation tree.

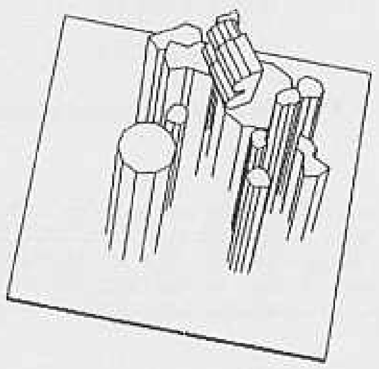


Figure 8g Obtained description.

Figure 8 (continued)

4. Grasp Planning and Pickup Motion

4.1. Legal Grasp Configuration and Collision-free Grasp Configuration

The grasp configuration should satisfy the following two conditions: [13]

1. It should produce a mechanically stable grasp, given the gripper's shape and the object's shape. Such configurations will be called *legal grasp configurations*.

2. The configuration must be achievable without collisions with other objects. Grasp configurations are limited by the relationship between the shape of the gripper and the shapes of neighboring obstacle. Such configurations will be called *collision-free grasp configurations*.

The first task is to find legal grasp configurations based on the gripper's shape and the object's shape. The following definition is used for the legal grasp configurations [13].

1. The object is not *translated* while the gripper is grasping the object.

2. The object is not *rotated* while the gripper is grasping the object.

In compile mode, possible legal grasp configurations are precomputed at each aspect from the silhouette of each representative attitude using the parallel-pair finding algorithm [13]. Basically, the object is treated as a 2D plate having the same shape as the silhouette at each representative attitude. Then, possible legal grasp point pairs to satisfy the parallel-pair condition are extracted. Point pairs on the boundary which have normals opposite to each other and are located on the line along the normals are extracted. From point pair positions and the plate normal, a legal grasp configuration is obtained with respect to the object centered coordinate system. These legal grasp configurations are stored at each representative attitude.

In run mode, the legal grasp configurations for each representative attitude stored in the compile mode are converted to corresponding legal grasp configuration of the object whose attitude and position are determined by the interpretation tree. This is because legal grasp configurations are describe with respect to the object centered coordinate system in the compile mode, while the obstacles and the object attitude are expressed in the world coordinate system. Applying the object transformation with respect to the world coordinate system to the legal grasp configurations give the legal grasp configurations with respect to the world coordinate system. Figure 9a shows the legal grasp configurations at the the object which belongs to the aspect S2.

Legal grasp configurations only describe the relationship between the gripper and the object grasped. Among these grasp configurations, we have to choose a grasp configuration that can be achieved without hitting other objects. This operation is done using an intersection-checking function of a geometric modeler. This function is usually available in a geometric modeler because it is necessary for executing a union operation between two objects.

Applying this function to our system requires the gripper work-space and the obstacles to be represented in the world of a geometrical modeler. Around the target region, there are a few regions which have been observed but have not been interpreted by the interpretation tree. These regions are obstacle regions for the picking-up motion. The neighboring regions are expressed as dodecahedral prisms in the world. The height of a prism agrees with the height of the corresponding region, and the cross section of the prism is an approximation of the region shape by a dodecagon. These dodecaheral prisms are used to determine a collision free configuration.

A gripper work-space, which is defined from a legal grasp configuration, must also be expressed in the world coordinate system. At each legal grasp configuration, a corresponding work-space prism of the finger is generated in the world of the geometrical modeler. This work-space prism is a union of the finger work space for the closing motion, and the finger work-space for the approaching motion along the direction of the finger to the object.

If this work-space prism intersects any one of the obstacle dodecahedral prisms, the finger would collide with the obstacle object while executing the pickup motion under this configuration. If this work-space prism intersects none of the obstacle prisms, then the legal grasp configuration corresponding to the work-space prism is a collision free configuration, and the gripper can pick up the object without collision under this configuration.

Among the possible collision free configurations, one collision free configuration is determined using an evaluation function. The nearest configuration to the mass center is chosen and is sent to the manipulator. Figure 9b shows the collision free configuration determined for the pickup motion.

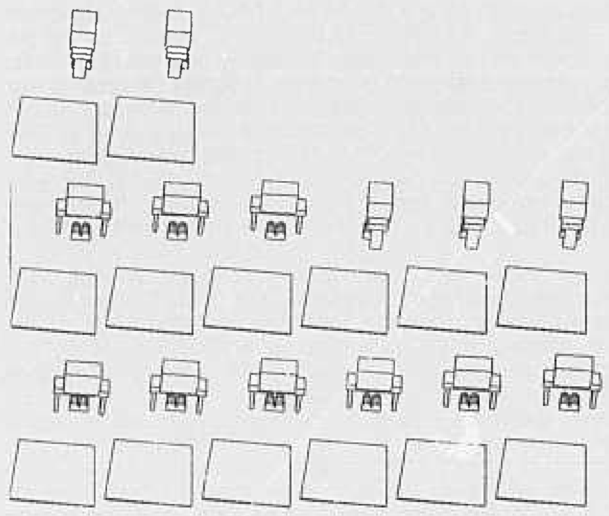


Figure 9a Legal grasp configurations at aspect S2.

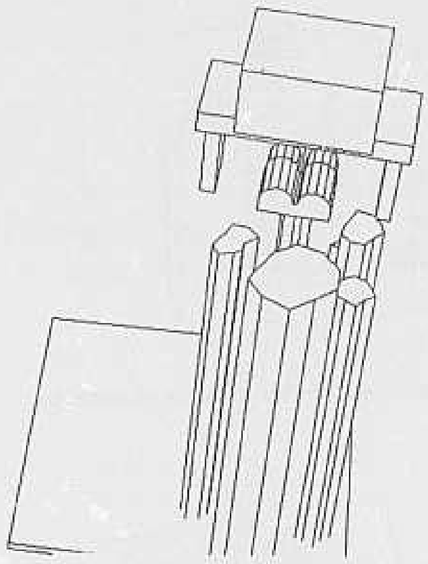


Figure 9b A collision free configuration.

Figure 9 Grasp planning.

4.2. Pickup motion

This system has four major coordinate systems: the eye coordinate system, the arm coordinate system, the model coordinate system, and the world coordinate system.

The object configuration is obtained in the eye coordinate system by the interpretation tree. The object dimensions are expressed in the model coordinate system. The legal grasp configurations are expressed in the model coordinate system. The model coordinate system and the world coordinate system are maintained in the geometrical modeler. Arm movements are executed in the arm coordinate system.

The object configuration obtained by the interpretation tree in the eye coordinate system is converted into the world coordinate system. The legal grasp configurations in the model coordinate system are converted in the world coordinate system based on the object configuration in the world coordinate system. The obstacles are also expressed in the world coordinate system. Thus, the collision-free configuration is obtained in the world coordinate system and must be converted into the arm coordinate system. The configuration in the arm coordinate system is sent to the arm for execution. The coefficients between coordinate systems are obtained as shown in Appendix II.

Figure 10 shows the execution process using the configuration in the arm coordinate system.

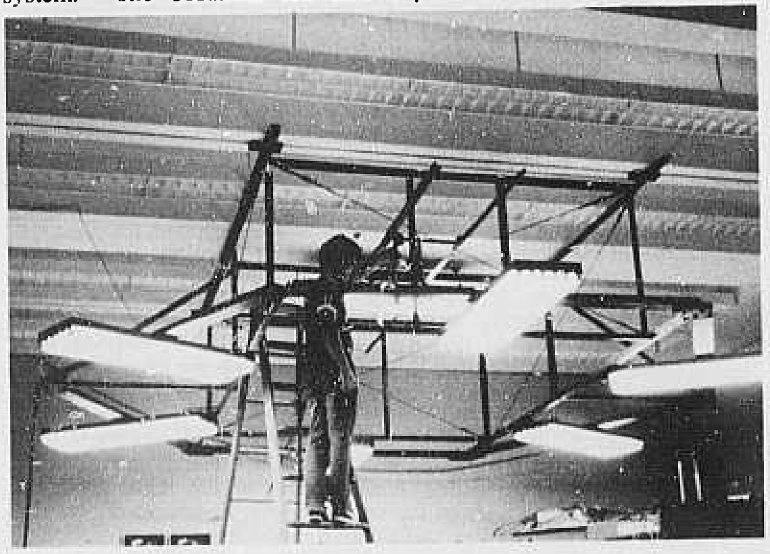


Figure 10a Light sources.

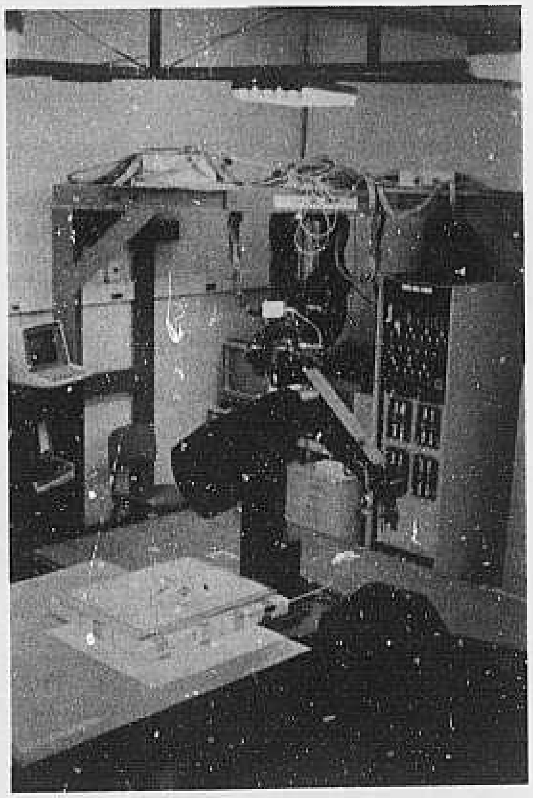


Figure 10b Puma arm.

Figure 10 Pickup motion.

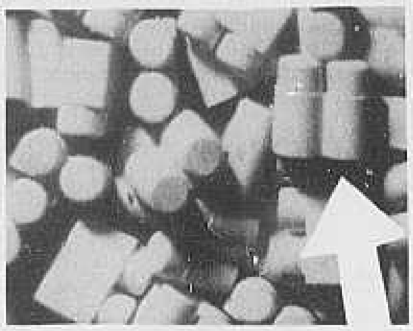


Figure 10c The target scene.

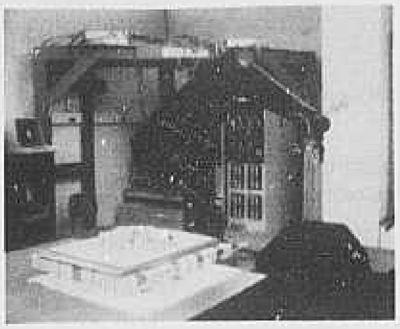


Figure 10d

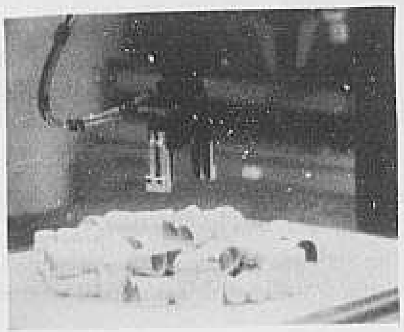


Figure 10e

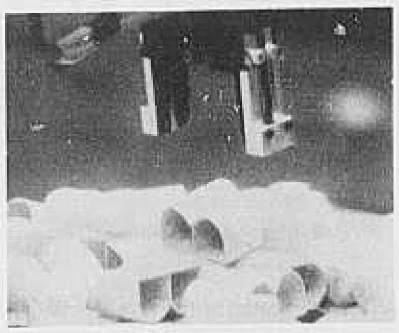


Figure 10f

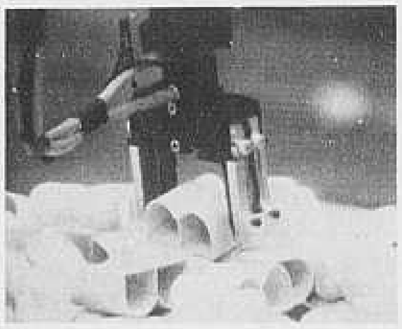


Figure 10g

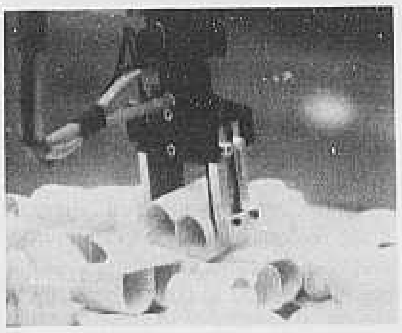


Figure 10h

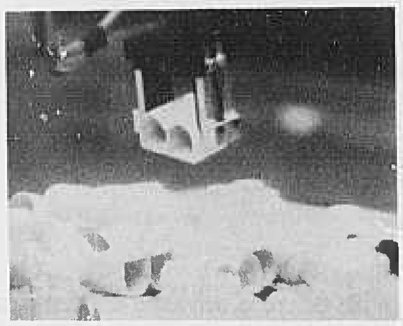


Figure 10i

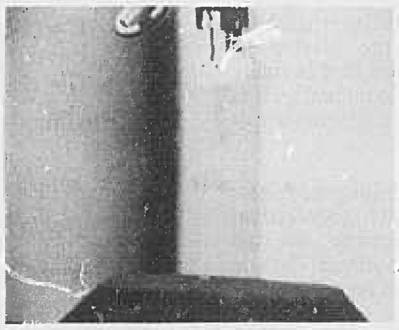


Figure 10j

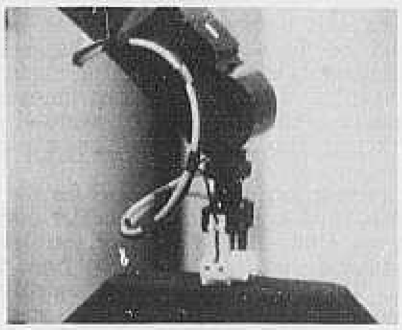


Figure 10k

Figure 10 (continued)

5. CONCLUDING REMARKS

This paper describes a vision system to localize an object by an interpretation tree.

This system has the following characteristics:

1. Aspects are derived from a geometrical modeler, automatically.
2. The interpretation tree controls the localization process to use the most appropriate features at each stage of the localization.
3. The obtained attitude and position are represented in the world model in a geometrical modeler for further use.

This paper assumes that the low level operations are reliable, and does not emphasize backtracking. This assumption works well in our situation because

1. The interpretation tree only analyzes the highest region, which is usually not occluded and exhibits all information necessary to be recognized.
2. The interpretation tree only uses the most reliable features at each matching stage.
3. The interpretation tree also contains some of the verification process and returns the target region as unrecognized. Thus, if the interpretation failed to verify the target region, the region is discarded and the second highest region is given to the interpretation tree by a higher-level flow controller. This iteration is repeated until one of the regions passes the examination.

However, an active backtracking schema would be necessary to apply this method to the analysis of occluded objects and to increase the efficiency of the interpretation process. Certainly, the next step is to explore how to include backtracking control in the interpretation tree.

This paper develops a flexible interpretation by an interpretation tree using multiple sensory inputs. Recent work in image understanding has led to techniques for computing surface orientation or surface depth. We can take various sensory inputs from the same scene by these methods. Since each technique has some merits and faults, we have to select one appropriate feature among many available features in each processing stage. This paper proposes to use the interpretation tree for this purpose. This flexible interpretation matching should be further explored. Right now the choice of discriminators used at nodes of the interpretation tree is made by "hand". In order to choose discriminators automatically, it is necessary to measure the uncertainty of each discriminator at each stage. This direction should be explored.

A geometrical modeler is used for the recognition problem. Models from a geometrical modeler possess rich geometrical features. Unfortunately however, the distance between the rich information and the information from the observed data is great. This paper uses the work models and the representative attitude to interface them. Effort is required to explore more convenient forms and methods to connect them.

The task of a vision system is to generate a description of the outer world. Some of the representations are symbolic, others use mathematic representations such as extended Gaussian images and generalized cylinders [36-38]. However, since the representation is needed for manipulation by other modules such as planning and navigation, the representation must be easy to manipulate [39]. This paper proposes to represent the outer world in the geometrical model, because a geometrical model representation is an easy basis to achieving further tasks. Certainly there are many path finding programs that start from the polyhedral representations [40]. How to express the outer world in such a representation should be explored more.

APPENDIX

I. Work Model

Original Face Inertia

One work model is the original face inertia. The original face inertia gives the rough shape information of a face. In order to obtain the inertia, we have to convert a needle map into a binary map. Here, the binary map has 1 at each pixel where the surface orientation can be obtained, and 0 at each pixel where the surface orientation cannot be obtained. The obtained binary map is represented as $m(x,y)$. From this $m(x,y)$ and the affine matrix T,

$$I_{xx}=\int m(x',y')dx'dx'$$

$$I_{xy}=\int m(x',y')dx'dy'$$

$$I_{yy}=\int m(x',y')dy'dy'$$

where

$$\begin{pmatrix}x'\\y'\end{pmatrix}=T\begin{pmatrix}x-\bar{x}\\y-\bar{y}\end{pmatrix}$$

and $(\bar{x},\bar{y})$ is the observed mass center of the face. From these I_{xx}, I_{xy}, I_{yy} we can determine the maximum inertia I_{max} and the direction α as follows:

$$I_{max}=(I_{xx}+I_{yy}+\sqrt{(I_{xx}+I_{yy})^2-4(I_{xx}I_{yy}-I_{xy}I_{xy})})/2$$
$$\alpha=(\tan^{-1}\{(2\,I_{xy})/(I_{xx}-I_{yy})\})/2$$

Surface Characteristic Distribution

Let us denote surface orientation as (p,q), where $p=z_x$ and $q=z_y$. Then, the first fundamental forms E,F,G are

$$E=(1+p^2)$$
$$F=pq$$
$$G=(1+q^2).$$

The second fundamental forms e,f,g are

$$e=p_x/\sqrt{1+p^2+q^2}$$
$$f=p_y/\sqrt{1+p^2+q^2}$$
$$g=q_y/\sqrt{1+p^2+q^2}$$

These coefficients give the Gaussian curvature K and the mean curvature H of the surface as follows:

$$K=(eg-f^2)/(EG-F^2)$$
$$H=(1/2)((eG-2fF+gE)/(EG-F^2))$$

Gaussian curvature K and mean curvature H determine the surface characteristic as follows:

1. *K=0 and H=0 then planar surface*

2. *$K=0$ and $H\neq 0$ then cylindrical surface*

3. *$K>0$ and $H>0$ then convex elliptic surface*

4. *$K>0$ and $H<0$ then concave elliptic surface*

5. *$K<0$ then hyperbolic surface*

The surface characteristic distribution is stored at each representative attitude. A subregion is generated based on a surface characteristic, and described by the surface characteristic and the rectangular existence area whose vertices are referenced to the coordinate of the mass center and the maximum inertia direction. When processing an image, the vertex positions are converted to image plane coordinates using the affine matrix. Then, the corresponding area is examined to determine whether surface patches having the characteristic exist or not.

II. Calibration between Coordinate Systems

Eye to World

We need coefficients from the eye coordinate system to the world coordinate system. These coefficients are decoupled into two groups and are obtained independently. The first group consists of the coefficients from the left and right TV camera coordinate systems to the world coordinate system. These coefficients will be used to get depth values at observed points. The second group consists of the coefficients from (x,y) of the left TV camera coordinate system and z of the world coordinate system into (x,y) of the world coordinate system. This is possible because the left TV camera's image plane is parallel to the x–y plane of the world coordinate system. Figure A.1 gives the relationship between coordinate systems. $\{X_w|x_w,y_w,z_w\}$ denotes the world coordinate system. The left and right TV cameras have their coordinate systems, $\{X_i^l|x_i^l,y_i^l,z_i^l\}$ and $\{X_i^r|x_i^r,y_i^r,z_i^r\}$.

Two TV cameras z axes intersect at the origin of the world coordinate system. The right TV camera's image plane and, thus, z axis and x axis is rotated around the y axis and is translated in d from the left TV camera's origin.

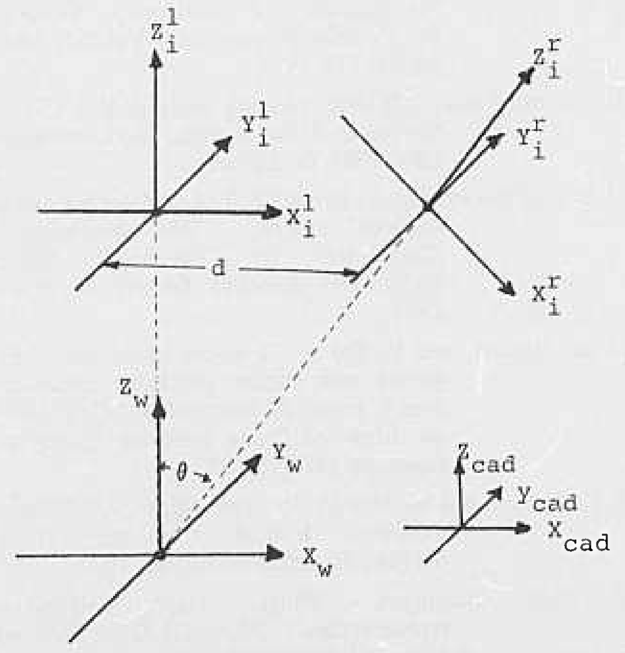


Figure A.1 The relationship between coordinate systems.

The camera transformation can be modeled as

$$u_i=\frac{x_i}{(1-z_i/f)},$$
$$v_i=\frac{y_i}{(1-z_i/f)},$$
$$w_i=\frac{z_i}{(1-z_i/f)}.$$

If the distance between the object and the TV camera is far compared to the forcal length, the camera model can be written as

$$u_i=\frac{xf}{z_i},\quad v_i=\frac{yf}{z_i},\quad w_i=f. \tag{1}$$

where u_i,v_i,w_i are defined as a projected point on the image plane of the spatial point X_i under the perspective projection.

Disparity to world

The coefficients from disparity value to z_w of the world coordinate system is necessary to measure the depth value of a spatial point from the left and right image points. We will consider only rotation of the right TV camera around y axis and the distance between two TV cameras, d, because we can easily set up two TV cameras to have their scan lines parallel with each other.

One spatial point is expressed as X^l in the left TV camera coordinate system and is also expressed as X^r in the right TV camera coordinate system. Thus,

$$X^l=TX^r, \tag{2}$$

where T

$$T=\begin{bmatrix} \cos\theta & 0 & \sin\theta & d \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

We can insert the camera model, Eq. (1) into Eq.(2) and solve Eq.(2) with with respect to z_i^l.

$$z_i^l=-d\cot\theta\left(1-\frac{u_r}{f_r}\tan\theta\right)\left(1-\left(\frac{u_l}{f_l}-\frac{u_r}{f_r}\cot\theta\right)+\frac{u_lu_r}{f_lf_r}\right)^{-1}$$

If we can assume that $1\ll\frac{u_lu_r}{f_lf_r}$, then

$$z_i^l=-d\cot\theta\left(1-\frac{u_r}{f_r}\tan\theta+\left(\frac{u_l}{f_l}-\frac{u_r}{f_r}\right)\cot\theta\right)$$

Thus, from $z_i^l=-d\cot\theta+z_w$,

$$z_w=au_l+bu_r+c \tag{3}$$

where z_w is the height in the world coordinate system, and

$$a=\frac{d}{f_l}\cot^2\theta$$
$$b=\frac{d}{f_r\sin^2\theta}$$
$$c=-d\cot\theta.$$

Since a,b, and c are constant, we will get these a,b, and c using the least squares method based on Eq.(3).

left TV camera coordinate system to the world coordinate system

The world coordinate system and the left TV camera coordinate system can be expressed as

$$X_w = MX_i^l + P,$$

where X_w^l, M, P denote the world coordinate, rotation matrix from the left TV camera coordinate system to the world coordinate system, and the translation part, respectively.

Thus, the transformation can be

$$X_w = MA_i(z_i^l/f^l) + P,$$

where

$$A_i = \begin{bmatrix} u_i^l \\ v_i^l \\ f^l \end{bmatrix}$$

This equation means that if calibration points are set on the plane parallel to the image plane, then z_i^l/f^l becomes constant on the plane and coefficients M can be obtained using a least square method.

More precisely, the least squares method gives Mz_i^l/f^l. This depends on z_i^l. Once this Mz_i^l/f^l is obtained at $z_i^l=z_0$ and $z_i^l=z_1$, the value at an intermediate point can be obtained using an interpolation method, while z_i^l is come from the disparity value.

ACKNOWLEDGMENT

Thanks go to Mark DeLouis who constructed the lighting stage, PUMA table, the fingers, and general setup.

The author thanks Takeo Kanade, Richard Mickelsen, Steve A. Shafer, Ellen Walker and the members of Image-Understanding-System Group of Carnegie-Mellon University for their valuable comments and discussions.

REFERENCES

[1] S. Tsuji and A. Nakamura, "Recognition of an object in a stack of industrial parts," in *Proc. 4th International Joint Conference on Artificial Intelligence*, pp.881-818, 1975.

[2] S. Tsuji and F. Matsumoto, "Detection of elliptic and linear edges by searching two parameter space," in *Proc. 5th International Joint Conference on Artificial Intelligence*, pp.569-575, 1977.

[3] M. Yachida and S. Tsuji, "A machine learning capability", in *Proc. 4th International Joint Conference on Artificial Intelligence*, pp.819-826, 1975.

[4] M.L. Baird, "Image segmentation technique for locating automotive parts on belt conveyers", in *Proc. 5th International Conference on Artificial Intelligence*, pp.694-695, 1977.

[5] W.A. Perkins, "Model-based vision system for scene containing multiple parts", in *Proc. 5th International Joint Conference on Artificial Intelligence*, pp.678-684, 1977.

[6] R. Bolles and R.A. Cain, "Recognizing and locating partially visible objects: the local-feature-focus method ," *J. Robotics Research*, Vol. 1, No. 3, pp.57-82, 1982.

[7] Y. Fukada, "Recognition of structural industrial parts stacked in bin", *Robotica*, vol. 2, 1984.

[8] C. Goad, "Special purpose automatic programming for 3D model-based vision", in *Proc. Image Understanding Workshop*, pp.94-104, 1983.

[9] N. Ayache, B. Faverjon, J. Boissonnat, and B. Bollack, "Automatic handling of overlapping workpieces", in *Proc. International Conference on Pattern Recognition 84*, pp.837-839, 1984.

[10] W.E.L. Grimson, and T. Lozano-Perez, "Model-based recognition and localization from sparse range or tactile data," *International Journal of Robotics Research*, Vol. 3, No. 3, 1984.

[11] J.R. Birk, R.B. Kelly, and H.A.S. Martines, "An orienting robot for feeding workpieces stored in bins ", *IEEE Trans.*, Vol. SMC-11, No.2, pp.151-160, 1981.

[12] K. Ikeuchi, B.K.P. Horn, S. Nagata, T. Callahan, and O. Feingold, "Picking up an object from a pile of objects", in *1st International Symposium on Robotics Research*, M. Brady and R. Paul (eds.) MIT Press, Cambridge, MA, 1984.

[13] K. Ikeuchi, H.K. Nishihara, B.K.P. Horn, P. Sobalvarro, and S. Nagata, "Determining grasp points using photometric stereo and the PRISM binocular stereo system", *J. Robotics Research*, Vol. 5, No. 1, pp.46-65, 1986.

[14] B.K.P. Horn, and K. Ikeuchi, "The Mechanical Manipulation of Randomly Oriented Parts", *Scientific American*, Vol.251, No.2, pp.100-111, 1984.

[15] K. Koshikawa, *SOLVER reference manual*, RM-85-33J, Computer Vision Section, Electrotechnical Lab., 1984, (in Japanese)

[16] K. Koshikawa, and Y. Shirai, "A 3-D modeler for vision research", *Proc. '85 International Conference on Advanced Robot,* pp.185-190, Robotics Society of Japan, 1985.

[17] M. Oshima, and Y. Shirai, "A model based vision for scenes with stacked polyhedra using 3D data ", *Proc. '85 International Conference on Advanced Robot,* Robotics Society of Japan, pp.191-198, 1985.

[18] F. Kimura, and M. Hosaka, *Program Package GEOMAP Reference Manual*, Computer Vision Section, Electrotechnical Lab, 1977.

[19] B.G. Baumgart, "Winged edge polyhedron representation", *STAN-CS-320*, Stanford Univ. A.I. Laboratory, 1972.

[20] I. Chakravarty, and H. Freeman, "Characteristic views as a basis for three-dimensional object recognition", in *Proc. The Society for Photo-Optical Instrumentation Engineers Conference on Robot Vision*, Vol. 336, SPIE, Bellingham, Wash., pp. 37-45, 1982.

[21] J.J. Koenderink, and A.J. Van Doom, "Internal representation of solid shape with respect to vision", *Biological Cybernetics*, Vol. 32, No. 4, pp. 211-216, 1979.

[22] K. Sugihara, " Automatic construction of junction dictionaries and their exploitation for analysis for range data", in *Proc. 6th International Joint Conference on Artificial Intelligence*, pp.859-864, 1979.

[23] C. Thorpe, and S. Shafer, "Correspondence in Linc Drawings of Multiple Views," in *Proc. 8th International Joint Conference on Artificial intelligence*, pp.959-965.

[24] M. Herman, "Matching three-dimensional symbolic descriptions obtained from multiple views", in *Proc. IEEE Computer Society Conference on Computer Vision and Patter Recognition*, San Francisco, June, 1985.

[25] M. Hebert and T. Kanade, "The 3D Profile method for object recognition", in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Francisco, June 1985.

[26] C.M. Brown, "Fast display of well-tessellated surface ", *Computer and Graphics*, Vol. 4, No. 2., pp. 77-85, 1979.

[27] D. Smith, "Using enhanced spherical images", *AI Memo 451*, MIT Artificial Intelligence Laboratory, 1979.

[28] B.K.P. Horn, "Extended Gaussian images", *Proc. of the IEEE*, Vol.72, No.12, pp.1671-1686, 1984.

[29] K. Ikeuchi, "Recognition of 3-D objects using the extended Gaussian image", *Proc. 7th International Joint Conference on Artificial Intelligence*, pp.595-600, 1981.

[30] K. Ikeuchi, "Determining attitude of object from needle map using extended Gaussian image", *AI memo No. 714*, MIT Artificial Intelligence Laboratory, Cambridge, MA, 1983.

[31] P. Brou, "Using the Gaussian image to find the orientation of object", *The International Journal of Robotics Research*, Vol.3, No.4, pp.89-125, 1983.

[32] J.J. Little, "Determining object attitude from extended Gaussian images", in *Proc. of Intern. Joint Conf. on Artificial Intelligence"*, pp.960-963, 1985.

[33] M. Brady, J. Ponce, A. Yuille, and H. Asada, "Describing surfaces", *Proc. 2nd International Symposium on Robotics Research*, H. Hanafusa and H. Inoue (eds.) MIT Press, Cambridge, MA, 1985.

[34] P.J. Besl, and R.C. Jain, "Intrinsic and extrinsic surface characteristics", *Proc. Computer Vision and Pattern Recognition Conference*, IEEE, San Francisco, pp. 226-233, 1985.

[35] K. Ikeuchi, "Determining a depth map using a dual photometric stereo", *J. Robotics Research*, Vol.6,No.1, 1987.

[36] T.O. Binford, "Visual perception by computer", in *Proc. IEEE Systems Science and Cybernetics Conf.*, 1971.

[37] R.A. Brooks, "Symbolic reasoning among 3-D models and 2-D images", *Artificial Intelligence*, Vol.17, Nos. 1-3, pp.285-348, 1981.

[38] S.A. Shafer, and T. Kanade, "The Theory of straight homogeneous generalized cylinder and a taxonomy of generalized cylinders," Carnegie-Mellon University Computer Science Department, Pittsburgh, Pa, CMU-CS-83-105, 1983.

[39] M. Herman, and T. Kanade, "The 3D MOSAIC scene understanding system: incremental reconstruction of 3D scene from complex images", *CMU CS Report*, CMU-CS-84-102, 1984.

[40] T. Lozano-Perez, "Automatic Planning of Manipulator Transfer Movements", *IEEE Trans. Sys. Man. Cyb.*, Vol.SMC-11, No.10, pp.681-689, 1981.

[41] R.J. Woodham, "Reflectance Map Techniques for Analyzing Surface Defects in Metal Castings," MIT Artif. Intell. Lab., MIT, *AI-TR-457*, 1978.

ANALYTICAL PROPERTIES OF GENERALIZED CYLINDERS AND THEIR PROJECTIONS

Jean Ponce, David Chelberg, Wallace Mann

Stanford Artificial Intelligence Laboratory
Stanford University, Ca 94305
January 14, 1987

Abstract: In this paper, we study the contours of generalized cylinders under orthographic projection. We first study the three dimensional contour generators: limbs and edges. We derive the general limb equation for generalized cylinders with a straight or curved spine. We solve this equation for three classes of generalized cylinders: solids of revolution, straight homogeneous generalized cylinders whose sweeping rule is a polynomial of degree less than or equal to 5, and generalized cylinders with a curved spine and a constant circular cross section. We then study the two dimensional contours, which are the projections in the image of the contour generators. For each of the above three classes, we use the limb equation to prove properties of the contours of generalized cylinders which are invariant with respect to the viewing direction. We finally use these properties in two implemented algorithms for finding the axis of straight homogeneous generalized cylinders.

Introduction

A *generalized cylinder* [1] is the solid obtained by sweeping a planar surface, its *cross section*, along a space curve, its *axis*, or *spine*. The axis is not necessarily straight, or even planar; the cross section is not necessarily circular or even constant; its deformation is governed by a *sweeping rule*. Generalized cylinders have been extensively used to represent 3D objects in computer vision [4],[9]–[16]. The most successful vision system to date using generalized cylinders as its primary representation for three dimensional objects is probably *Acronym* [4]. Even in Acronym however, only very restricted sub-classes of generalized cylinders have been implemented (circular or simple polygonal cross section, straight or circular spine, linear or bilinear sweeping rule). These relatively simple generalized cylinders project onto relatively simple 2D shapes, namely *ribbons* and *ellipses* [4].

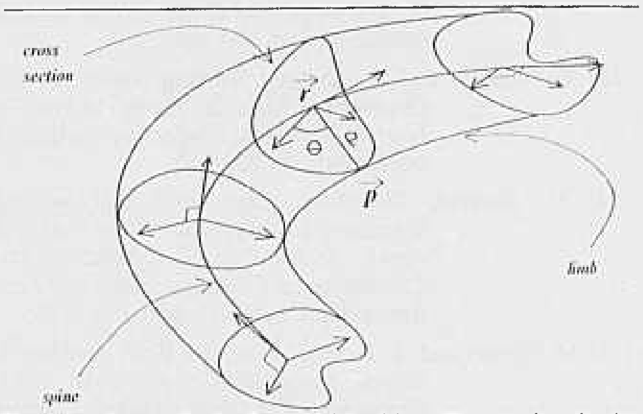


Figure 1. A generalized cylinder is defined by a 3D curve, its spine (or axis), and planar cross sections orthogonal to the spine. The cross sections are defined in polar coordinates.

To develop a vision system using a richer class of generalized cylinders, we need to choose for these primitives a set of *features* which are observable in the image, and to find a set of *properties* of these features which are invariant (or quasi invariant) with respect to the viewing direction. Image discontinuities can be caused by illuminance and reflectance discontinuities, or by the geometry of the viewed objects themselves. We call the latter "geometric" discontinuities *contours* in this paper, and use them as our set of observable features. They can be found in an image through edge detection.

As shown in [2], the contours are the projections of two kinds of *contour generators*, namely the *limbs*, where the surface turns away from the viewer, and the *edges*, where the surface orientation is discontinuous. Both limbs and edges are 3D curves drawn on a surface. On a smooth surface, a point belongs to a limb iff the normal at this point is orthogonal to the viewing direction. If the surface contains edges, an edge point is also a limb point iff the dot product of the normal with the viewing direction has different signs on each side of the edge.

Support for this work was provided in part by the Air Force Office of Scientific Research under contract F33615-85-C-5106 and by the Advanced Research Projects Agency of the Department of Defense under Knowledge Based Vision contract AIADS S1093-S-1. Wallace Mann was partially supported by a fellowship from the American Electronics Association, F.M.C., Corp., Faculty Development Program.

To study the properties of the contours, we first study the contour generators, then deduce properties of their projections. In 3D space, edges are relatively simple, because they are curves physically drawn on the surface and are independent of the viewing direction. On the contrary, limbs are only defined with respect to the viewing direction. The properties of surface limbs have been studied in [6], and for the case of generalized cylinders, in [9],[12],[14]–[16]. Here, we use an approach very similar to Shafer's and Kanade's [15] study of the properties of straight homogeneous generalized cylinders.

To study the limbs, we first derive the limb equation for the general case of straight generalized cylinders and generalized cylinders with a 3D spine (Section 1). We solve this equation for three classes of generalized cylinders (Section 2): solids of revolution (SR's), straight homogeneous generalized cylinders (SHGC's) whose sweeping rule is a polynomial of degree less than or equal to 5, and generalized cylinders with a 3D spine and a constant circular cross section (CCGC's). We use the limb equation to prove a set of invariant properties for the contours of these three classes of generalized cylinders.

We first give some general results for the projections of generalized cylinders and prove that the contours of SR's and CCGC's are symmetric with respect to the projection of the image of their axis (section 3). We then show three invariant properties of the contours of SHGC's. We prove that for a given cross section, the tangents to the contours intersect on the image of the axis (section 4). We then prove in section 5 that extrema of the distance between the contours and the image of the axis correspond to extrema of the sweeping rule. Finally we prove in section 6 that zeros of curvature of the contours correspond to zeros of curvature of the sweeping rule. These three results are true for an arbitrary scaling sweeping rule.

In the last section, we use these three properties in two simple algorithms for finding the axis of a straight homogeneous generalized cylinder in an image. We demonstrate these algorithms on real images.

1. Deriving the limb equation

1-1. Straight generalized cylinders

In this section, we derive the general limb equation for straight generalized cylinders. Although generalized cylinders are volumes, the limbs are defined in terms of their surface, so we use a parametric representation of these surfaces. In all the sequel, we restrict our attention to generalized cylinders for which each cross section is planar, orthogonal to the axis, and can be represented by a real function $\rho(\theta)$ in some polar coordinate system of its plane. Note that this implies that the cross section is star shaped (Figure 2) with respect to some point which is the polar coordinate system origin. Convex cross sections are a strict sub-class of star shaped cross sections. If $(O, \vec{\imath}, \vec{\jmath}, \vec{k})$ is an orthonormal reference frame, a straight generalized cylinder of axis $\vec{k}$ can then be written, without loss of generality

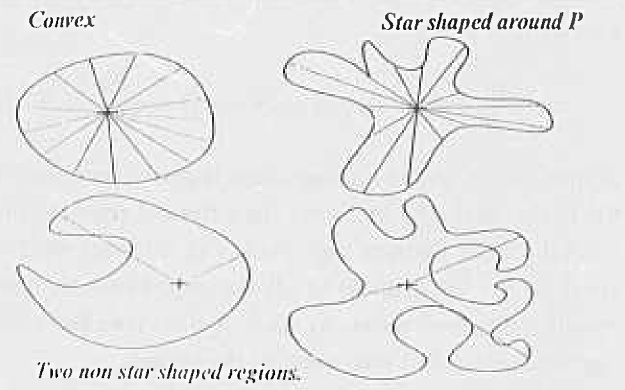


Figure 2. A region is star shaped with respect to a point P iff for each point on its boundary, the whole segment between P and this point is included inside the region.

$$\overrightarrow{OP}(z,\theta) = u(z,\theta)(\cos\theta\vec{\imath} + \sin\theta\vec{\jmath}) + z\vec{k}$$

The function $z \rightarrow u(z,\cdot)$ which associates to each z the function defining the corresponding cross section is the sweeping rule of the generalized cylinder. u is supposed to be strictly positive and differentiable in this section.

The normal to the surface is given by the vector product of the partial derivatives of $\overrightarrow{OP}$ with respect to θ and z. Let us write these partial derivatives (the arguments z and θ are omitted in the sequel)

$$\frac{\partial\overrightarrow{OP}}{\partial\theta} = (\frac{\partial u}{\partial\theta}\cos\theta - u\sin\theta)\vec{\imath} + (\frac{\partial u}{\partial\theta}\sin\theta + u\cos\theta)\vec{\jmath}$$

$$\frac{\partial\overrightarrow{OP}}{\partial z} = \frac{\partial u}{\partial z}\cos\theta\vec{\imath} + \frac{\partial u}{\partial z}\sin\theta\vec{\jmath} + \vec{k}$$

It follows that the normal $\vec{n}$ to the surface is given by

$$\vec{n} = (\frac{\partial u}{\partial\theta}\sin\theta + u\cos\theta)\vec{\imath} + (-\frac{\partial u}{\partial\theta}\cos\theta + u\sin\theta)\vec{\jmath} - u\frac{\partial u}{\partial z}\vec{k}$$

The limbs are given by the equation $\vec{v}\cdot\vec{n} = 0$, where $\vec{v}$ is the viewing direction. In the case of orthographic projection, $\vec{v}$ is a constant unit vector. If its spherical coordinates are (α, β) in the frame $(\vec{\imath}, \vec{\jmath}, \vec{k})$, we have

$$\vec{v} = \sin\beta(\cos\alpha\vec{\imath} + \sin\alpha\vec{\jmath}) + \cos\beta\vec{k}$$

Writing that $\vec{v}\cdot\vec{n} = 0$, we get the limb equation for straight generalized cylinders:

$$\sin\beta\left(\frac{\partial u}{\partial\theta}\sin(\theta-\alpha) + u\cos(\theta-\alpha)\right) = u\frac{\partial u}{\partial z}\cos\beta \quad (1)$$

A special case for the limb equation is *polar views* ($\sin\beta = 0$). In this case, the limbs are the extremal cross sections (i.e. the cross sections such that u is extremal with respect to z). We suppose in all the sequel that the view considered is non polar, so $\sin\beta \neq 0$. Views for which $\cos\beta = 0$ are called *side views* in the sequel.

1-2. Generalized cylinders with a curved axis

Consider now a 3D curve Γ (Figure 3). If Γ is defined by the vector $\overrightarrow{OR}(s)$, where s is the arc length along Γ, then the Frénet frame $(\vec{t}, \vec{n}, \vec{b})$ is defined by

$$\frac{d\overrightarrow{OR}}{ds} = \vec{t},\ \frac{d\vec{t}}{ds} = \kappa\vec{n},\ \frac{d\vec{n}}{ds} = -\kappa\vec{t} + \tau\vec{b},\ \frac{d\vec{b}}{ds} = -\tau\vec{n}$$

$\vec{t}$, $\vec{n}$, and $\vec{b}$ form an orthonormal frame. κ and τ are respectively the curvature and the torsion of Γ. $\vec{t}$, $\vec{n}$, $\vec{b}$, κ and τ are all functions of s. A generalized cylinder of spine Γ can be written, without loss of generality

$$\overrightarrow{OP}(s,\theta) = \overrightarrow{OR}(s) + u(s,\theta)(\cos\theta\vec{n} + \sin\theta\vec{b})$$

We can express the normal $\vec{n}$ to the surface by the vector product of the two partial derivatives. They are given by

$$\frac{\partial\overrightarrow{OP}}{\partial\theta} = \left(\frac{\partial u}{\partial\theta}\cos\theta - u\sin\theta\right)\vec{n} + \left(\frac{\partial u}{\partial\theta}\sin\theta + u\cos\theta\right)\vec{b}$$

$$\frac{\partial\overrightarrow{OP}}{\partial s} = (1-\kappa u\cos\theta)\vec{t} + \left(\frac{\partial u}{\partial s}\cos\theta - u\tau\sin\theta\right)\vec{n} + \left(\frac{\partial u}{\partial s}\sin\theta + u\tau\cos\theta\right)\vec{b}$$

This yields the following value for $\vec{n}$

$$\vec{n} = u\left(\tau\frac{\partial u}{\partial\theta} - \frac{\partial u}{\partial s}\right)\vec{t} + (1-\kappa u\cos\theta) \times \left[\left(\frac{\partial u}{\partial\theta}\sin\theta + u\cos\theta\right)\vec{n} + \left(-\frac{\partial u}{\partial\theta}\cos\theta + u\sin\theta\right)\vec{b}\right]$$

If α and β are the spherical coordinates of $\vec{v}$ in $(\vec{n}, \vec{b}, \vec{t})$ (α and β are functions of s), we have

$$\vec{v} = \cos\beta\vec{t} + \sin\beta(\cos\theta\vec{n} + \sin\theta\vec{b})$$

And the limb equation for generalized cylinders with a curved spine can be rewritten:

$$\sin\beta(1-\kappa u\cos\theta)\left(\frac{\partial u}{\partial\theta}\sin(\theta-\alpha) + u\cos(\theta-\alpha)\right) = -\cos\beta\, u\left(\tau\frac{\partial u}{\partial\theta} - \frac{\partial u}{\partial s}\right) \quad (2)$$

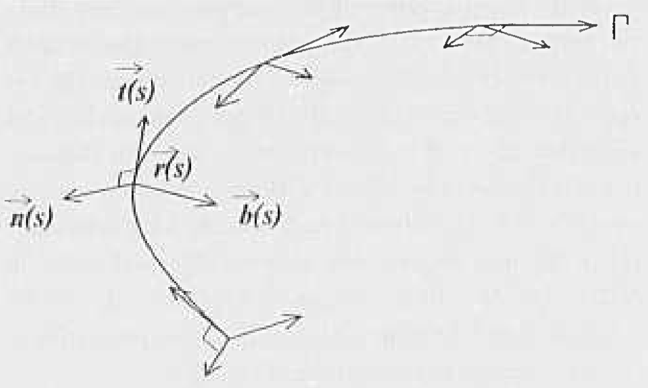


Figure 3. The Frénet frame associated to a 3D curve. $\vec{t}$ is the tangent to the curve, and it forms with the normal $\vec{n}$ and the binormal $\vec{b}$ a moving orthonormal frame which is function of the arc length s along the curve.

2. Solving the limb equation

2-1. Solids of revolution

A solid of revolution is obtained by rotating a planar curve around a straight line. Here we restrict our attention to the case where this generating curve can be expressed as a strictly positive function r of z In this case $u(z,\theta)$ is identical to $r(z)$. Its partial derivative with respect to θ is identically 0, and its derivative with respect to z is $r'(z)$. Substituting in Equation (1), and dividing by r, we finally get

$$\sin\beta\cos(\theta-\alpha) = r'(z)\cos\beta \quad (3)$$

Except in polar views, the limbs of a SR, if they exist, have an analytical expression $\theta = f(z)$. See [15] for a slightly different approach, leading to the same result. Notice finally that the limbs of a surface of revolution are in general not planar (although for example the limbs of a cone are straight lines). Figure 4 shows the limbs found for two views of a solid of revolution.

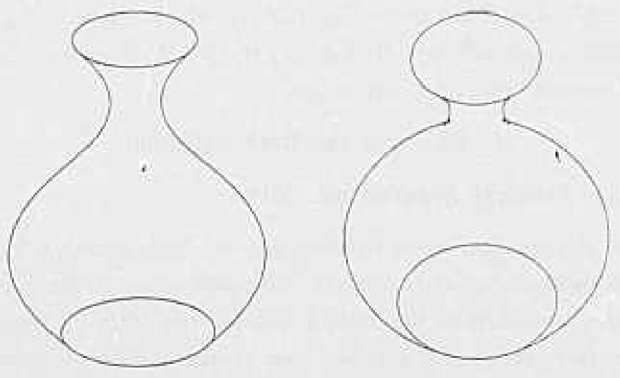


Figure 4. Two different views of a bottle-shaped solid of revolution.

2-2. Straight homogeneous generalized cylinders

We now consider straight homogeneous generalized cylinders (following the taxonomy of [15]): they are generalized cylinders whose cross sections are scaled versions of a reference curve. In this case the function $u(z,\theta)$ can be decoupled into the product $r(z)\rho(\theta)$, where ρ specifies the reference cross section, and r the scaling function. The partial derivatives of u are $\frac{\partial u}{\partial \theta} = r\rho'$ and $\frac{\partial u}{\partial z} = r'\rho$. By substituting in Equation (1), and dividing by $r\rho^2$ we get

$$\sin\beta[\frac{\rho'}{\rho^2}\sin(\theta-\alpha) + \frac{1}{\rho}\cos(\theta-\alpha)] = r'\cos\beta \qquad (4)$$

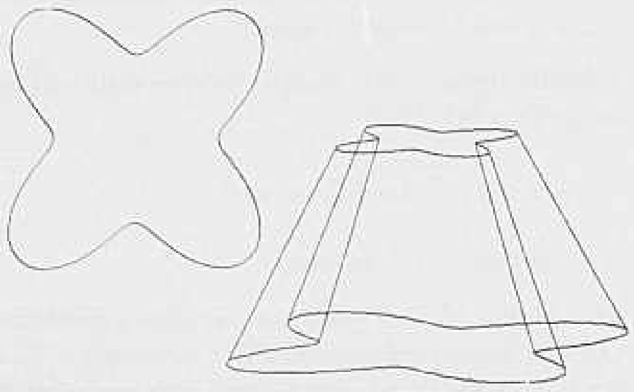


Figure 5. A star shaped cross section and a SHGC obtained by scaling linearly this cross section

In particular this implies that if the scaling function r is constant or linear, or if the view is a side view ($\cos\beta = 0$), the limbs, if they exist, are planar curves (they are given by a constant left member of the equation, which is a function of θ only, and can be found by looking for the zeros of a one parameter function). In the case of a polar view, the limbs are the cross sections corresponding to extrema of the r function. Finally, in the the remaining case, if the scaling function is a polynomial of degree between 2 and 5, the right member of the equation is a polynomial of degree between 1 and 4. As there exist closed form solutions for the roots of such polynomials, this gives an analytical expression for the limbs of the form $z = f(\theta)$. Arbitrary scaling functions can be approximated in terms of low order splines. Figures 5 and 6 show the limbs of two straight homogeneous generalized cylinders.

2-3. GC's with 3D spine and circular cross section

Consider now the special case of a generalized cylinder with a curved spine whose cross section is circular and constant. The function u is a constant ρ_0, and its two partial derivatives are identically 0. If we denote the radius of curvature of the spine by $R(s) = 1/\kappa(s)$, we get by substituting in Equation (2)

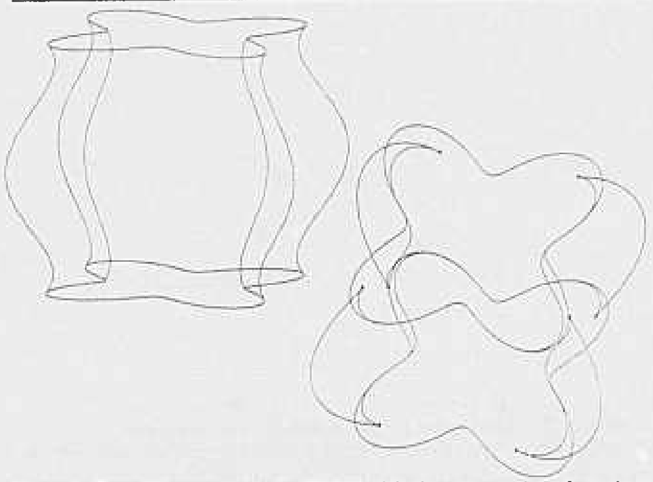


Figure 6. Two different views of a SHGC with the cross section of previous Figure, and a polynomial sweeping rule.

$$\sin\beta(s)(1-\frac{\rho_0}{R}\cos\theta)\rho_0\cos(\theta-\alpha(s)) = 0 \qquad (5)$$

We have $\rho_0 > 0$. Also, for the generalized cylinder to be well defined, we must have $R > \rho_0$ (otherwise the surface can self intersect). So a point is a limb iff $\beta(s) = 0$ (this is the local equivalent of a polar view; when the viewing direction is aligned with the tangent to the spine, the whole corresponding cross section is a limb) or $\theta = \alpha(s) \mp \pi/2$. In the latter case, the limb is given by an analytic 3D curve parameterized by s. Figures 7 and 8 show the limbs computed for two views of a torus and a helix.

3. The projections of generalized cylinders

All the previous results are given for limbs considered as three dimensional curves. The 2D projections of these curves, however, and not the 3D curves themselves, form observable contours in images. We now derive elementary properties of these projections.

3-1. Image coordinates of generalized cylinders

We first consider straight generalized cylinders. Equation (1) was given in the reference frame of the considered SHGC. To study the projections of the contour generators we introduce the new frame $(O,\vec{u},\vec{v},\vec{w})$, where $\vec{v}$ is the viewing direction, and $\vec{u}$ and $\vec{w}$ are the following unit vectors

$$\vec{u} = -\cos\beta(\cos\alpha\vec{\imath} + \sin\alpha\vec{\jmath}) + \sin\beta\vec{k}$$

$$\vec{w} = -\sin\alpha\vec{\imath} + \cos\alpha\vec{\jmath}$$

This frame is orthonormal; $(O,\vec{w},\vec{u})$ is a basis of the image plane, and $\vec{u}$ is the projection of the SHGC's axis in the image plane (Figure 9).

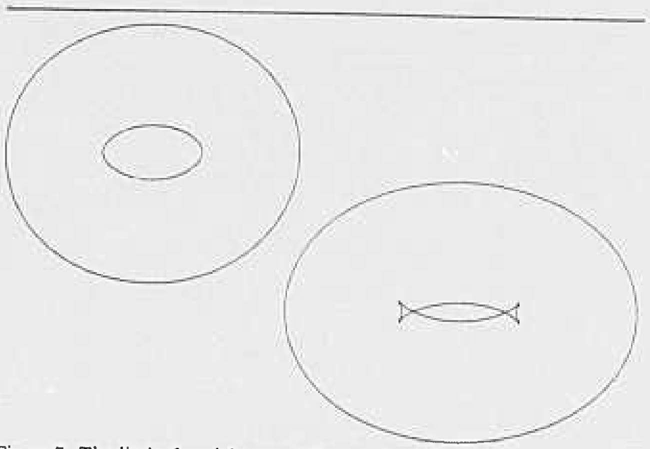


Figure 7. The limbs found for two different views of a torus.

The image coordinates of a point P are given by $x = \overrightarrow{OP} \cdot \vec{w}$ and $y = \overrightarrow{OP} \cdot \vec{u}$. If P belongs to the surface of a straight generalized cylinder (P is not necessarily a limb point), with the parameters z and θ, its coordinates are given by

$$x = u \sin(\theta - \alpha)$$

$$y = z \sin\beta - u \cos(\theta - \alpha) \cos\beta$$

For any view, the projection of the axis in the image is given by $x = 0$.

Consider now a generalized cylinder with a curved axis. For a given value of s we can define the orthonormal frame $(R(s), \vec{u}(s), \vec{v}(s), \vec{w}(s))$, where $\vec{v}$ is the viewing direction, and

$$\vec{u} = -\cos\beta(\cos\alpha\vec{n} + \sin\alpha\vec{b}) + \sin\beta\vec{t}$$

$$\vec{w} = -\sin\alpha\vec{n} + \cos\alpha\vec{b}$$

In this case, $\vec{u}$ is the projection in the image plane of the tangent to the axis. The coordinates of the point (s, θ) in the image plane basis $(R, \vec{w}, \vec{u})$ are

$$x = u \sin(\theta - \alpha)$$

$$y = -u \cos(\theta - \alpha) \cos\beta$$

3-2. A simple example: solids of revolution and generalized cylinders with a 3D spine and a constant circular cross section

We first consider the case of a solid of revolution. In this case, we have $u = r(z)$, and the coordinates of the projection of the point $P(z, \theta)$ are

$$x = r \sin(\theta - \alpha)$$

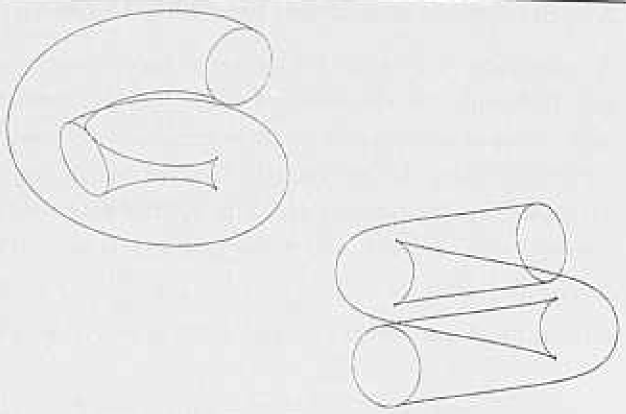


Figure 8. The limbs found for two different views of a helix.

$$y = z \sin\beta - r \cos(\theta - \alpha) \cos\beta$$

Using the limb equation, we can substitute $\cos(\theta - \alpha)$ and we get for a limb point

$$x = \epsilon r \sqrt{1 - r'^2 \cot^2\beta}; \quad \epsilon = \mp 1$$

$$y = z \sin\beta - rr' \cot\beta \cos\beta$$

For a given z, all limb points have the same y coordinate, and may have one of two opposite x coordinates. Thus, the projection of a SR is symmetric with respect to its axis.

Let us consider now the case of a CCGC. We can again substitute the expression of $\cos(\theta - \alpha)$ by using the limb equation. We get

$$x = \epsilon \rho_0; \quad \epsilon = \mp 1$$

$$y = 0$$

So the contours of a CCGC are symmetric with respect to the projection of the axis. Notice finally that the results presented in this section could be used to find the axis of a SR or a CCGC in images by using techniques analogous to Brady's and Asada's *smooth local symmetries* [3].

3-3. Image derivatives for a SHGC

We now restrict our attention to straight homogeneous generalized cylinders. We suppose in all the sequel that $r(z)$ is C^2 (twice continuously differentiable), and $\rho(\theta)$ is C^0 (continuous) and piecewise C^2. This includes straight homogeneous generalized cylinders with *edges* (curves on a SHGC which correspond to an orientation discontinuity of $\rho(\theta)$). The limb analysis of the previous sections holds everywhere except at edges. Therefore, limb points are supposed to be C^1 in all the sequel (the case of edge points is considered separately when necessary).

The limb equation is an implicit equation. In section 2, we have parameterized the limbs by θ. In all the sequel however, we are going to assume that a limb can be locally parameterized by z, i.e. that θ can be written as a continuously differentiable function of z. We now show under what conditions this assumption is valid.

The SHGC limb equation (4) can be rewritten

$$0 = F(z,\theta) = sin\beta f(\theta) - cos\beta g(z)$$

Where

$$f(\theta) = \frac{1}{\rho}(\theta)\cos(\theta-\alpha) + \frac{\rho'}{\rho^2}(\theta)\sin(\theta-\alpha)$$

$$g(z) = r'(z)$$

As both ρ and r are continuously differentiable the *implicit function theorem* (see for example [18]) shows that if $f'(\theta_0) \neq 0$ at some point θ_0, then θ can locally be parameterized as a C^1 function of z in a neighborhood of θ_0. We have

$$f'(\theta) = \frac{1}{\rho^3} sin(\theta-\alpha)[\rho\rho'' - \rho^2 - 2\rho'^2]$$

We suppose in the sequel that $f'(\theta) \neq 0$ everywhere, and we parameterize θ by z. To be completely rigorous, we should choose an other parameterization at the (typically few) points where $f'(\theta) = 0$. Again, this is not done for the sake of conciseness.

Finally, to derive the curvature of a limb, we suppose in the sequel that θ can be considered as a C^2 function of z except at cusp points. It is possible to show (see [12]) that cusp points are given by the equation

$$\rho^4 rr'' + (\rho\rho'' - \rho^2 - 2\rho'^2)\sin^2(\theta-\alpha)\tan^2\beta = 0 \quad (6)$$

A last remark: why not choose θ as a parameter instead of z? This is possible at any point such that $g'(z) = r''(z) \neq 0$. Unfortunately, zeros of curvature (see section 6) verify $r''(z) = 0$, and that is why we have chosen the z parameterization. The edges are also naturally parameterized by z.

4. The intersecting tangents lemma

Shafer and Kanade [15] have proved the following result:

3D intersecting tangents lemma: *For any two points with the same z value, the tangents to the surface in the direction of increasing z intersect on the axis.*

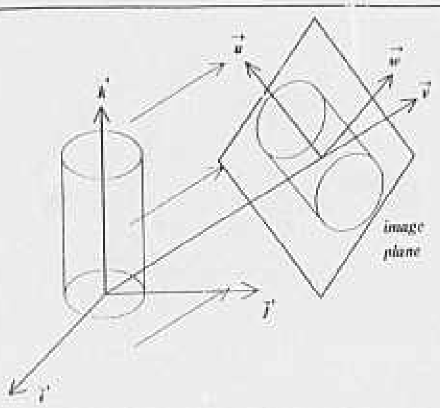


Figure 9. The coordinate systems used for the projections of generalized cylinders

This implies in particular that for any family of curves of constant θ drawn on the surface of a SHGC, the tangents intersect on the axis. An example of such curves is what Shafer and Kanade call the crease contours, i.e. for us the edges of a SHGC. Is it possible to extend this result to the other contour generators, i.e. the limbs of a SHGC? This is important as such a property could be used for finding the axis of a SHGC in an image (see section 7). We prove in this section that this lemma does not hold for limbs in three dimensions, but that the following 2D version of this result does hold (Figure 10).

2D intersecting tangents lemma: *In orthographic projection, for any two contour points with the same z value, the tangents to the contours intersect on the image of the axis.*

This lemma generalizes Shafer's and Kanade's result. We now prove it. We first give a short proof, analogous to Shafer's and Kanade's proof, that the 3D lemma is valid for curves of constant θ. This proves our lemma for side views and edges in general viewing conditions. We then show that the 3D lemma is false when θ is not constant. Finally, we show the 2D lemma for non side views.

4-1. Tangents to curves of constant θ

We consider a curve of constant $\theta = \theta_0$ drawn on the SHGC's surface and parameterized by z. A point on the curve can be written

$$\overrightarrow{OP}(z) = \rho(\theta_0)r(z)(\cos\theta_0\vec{\imath} + \sin\theta_0\vec{\jmath}) + z\vec{k}$$

This curve is planar. Its tangent is given by

$$\vec{t}(z) = \rho(\theta_0)r'(z)(\cos\theta_0\vec{\imath} + \sin\theta_0\vec{\jmath}) + \vec{k}$$

If we denote $\rho(\theta_0)$ by ρ_0, a point on the tangent line at $\overrightarrow{OP}$ is given by $\overrightarrow{OP} + k\vec{t}$.

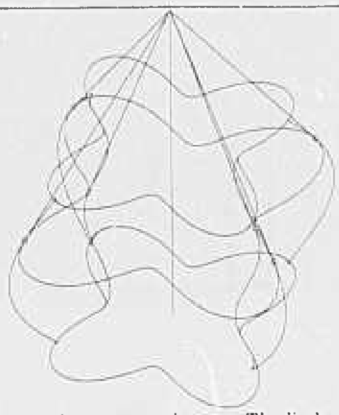


Figure 10. The intersecting tangents lemma. The limbs and tangents are computed analytically.

$$\overrightarrow{OP} + k\vec{t} = \rho_0(r(z) + kr'(z))(\cos\theta_0\vec{\imath} + \sin\theta_0\vec{\jmath}) + (z+k)\vec{k}$$

The tangent line intersects the axis for $k = -r/r'$ and the intersection point is given by

$$\overrightarrow{OP}_0 = (z - \frac{r}{r'}(z))\vec{k}$$

This point is independent of θ_0, so, for a given z, the tangents to all curves of constant θ intersect on the axis. This result applies in particular to limbs in side views, and edges in general viewing conditions.

We now show that the result is not true for a curve drawn on a SHGC which does not have a constant θ coordinate (a limb in oblique view is an example of such a curve). We suppose (as in the rest of the paper) that along this curve, θ can locally be parameterized as a C^1 function of z. The tangent to the curve is then

$$\vec{t} = [(\theta'\rho' r + \rho r')\cos\theta - \theta'\rho r\sin\theta]\vec{\imath}$$
$$+ [(\theta'\rho' r + \rho r')\sin\theta + \theta'\rho r\cos\theta]\vec{\jmath} + \vec{k}$$

A point on the tangent line $\overrightarrow{OP} + k\vec{t}$ is on the axis iff its coordinates along $\vec{\imath}$ and $\vec{\jmath}$ are 0, ie

$$[\rho r + k(\theta'\rho' r + \rho r')]\cos\theta - k\theta'\rho r\sin\theta = 0$$

and

$$[\rho r + k(\theta'\rho' r + \rho r')]\sin\theta + k\theta'\rho r\cos\theta = 0$$

This is equivalent to the two equations

$$\rho r + k(\theta'\rho' r + \rho r') = 0$$

$$k\theta'\rho r = 0$$

This system does not have a solution unless $\theta' = 0$ (in the second equation, $\theta' \neq 0$ implies that $k = 0$, but then the first equation implies $\rho r = 0$, which is impossible as ρ and r are strictly positive) in which case it is the same solution as for curves of constant θ.

4-2. Proof for the limbs in the orthographic case

A point is on the image of the SHGC axis iff $x = 0$. From now on, a point will be by default an image point, so "a limb point", or "a point on the axis" is in fact a point on the image of a limb, or a point on the image of the axis. To compute the tangent to a limb for a given z, let us assume again that for each limb segment, θ can be considered as a differentiable function of z. The tangent direction to a point (x, y) is then given by

$$x'(z) = (\theta'\rho' r + \rho r')\sin(\theta - \alpha) + \theta'\rho r\cos(\theta - \alpha)$$
$$y'(z) = \sin\beta - (\theta'\rho' r + \rho r')\cos\beta\cos(\theta - \alpha)$$
$$+ \theta'\rho r\cos\beta\sin(\theta - \alpha)$$

A point on the tangent line is given by

$$x_k = x + kx'; \; y_k = y + ky'$$

The intersection of the tangent and the axis is given by $x_k = 0$ or $k = -x/x'$, its y coordinate is

$$y_0 = y - x\frac{y'}{x'}$$

By substituting the x, y, x' and y' values and simplifying, we get

$$y_0 = z\sin\beta - \frac{\rho r}{x'}[\theta'\rho r\cos\beta + \sin(\theta - \alpha)\sin\beta]$$

We can rewrite x'

$$x' = \theta' r[\rho'\sin(\theta - \alpha) + \rho\cos(\theta - \alpha)] + \rho r'\sin(\theta - \alpha)$$

The above equations stand for the tangent to any curve drawn on a SHGC. We can simplify the expression of x' in the case of a limb point. Using the limb equation (4) we get

$$x' = \rho r'[\theta'\rho r\cot\beta + \sin(\theta - \alpha)]$$

So we finally get

$$y_0 = \left(z - \frac{r}{r'}(z)\right)\sin\beta$$

This value is independent of θ. It is equal to the y coordinate of the projection of the edges' intersection point, and the lemma is proved. Note that for extrema of r, the intersection is at an infinite distance along the axis.

5. The extrema of distance lemma

Suppose the axis has been found in the image. Is it then possible to relate the distance between contour points and the axis to the scaling function r? In fact, it is, and we now prove the following lemma (Figure 11)

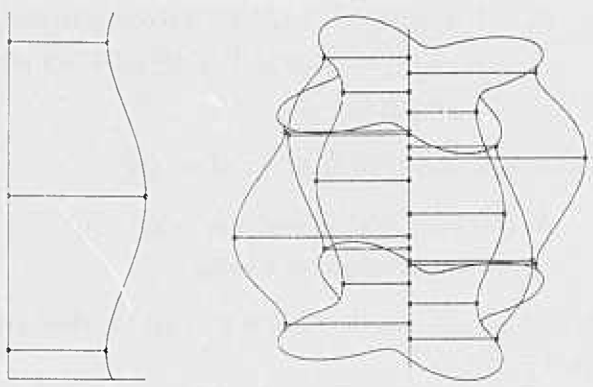


Figure 11. The extrema of the scaling sweeping rule and the corresponding extrema of distance between the axis and the limbs.

Extrema of distance lemma: *In orthographic projection, the distance between a contour and the image of the axis is extremal iff the point is a cusp point or its z coordinate corresponds to an extremum of the sweeping rule function r.*

The (signed) distance between a limb point and the axis is simply the x coordinate of the point, so this distance is extremal iff $x' = 0$. The proof for edges is straightforward, as for the edge $\theta = \theta_0$ we have

$$x'(z) = r'(z)\rho(\theta_0)sin(\theta_0 - \alpha)$$

We now prove the result for limb points. We have shown in the previous section that for limbs, we have

$$x' = \rho r'[\theta'\rho r \cot\beta + \sin(\theta - \alpha)]$$

It follows that $x' = 0$ iff $r' = 0$ (i.e. the point corresponds to an extremum of r), or the quantity $C = \theta'\rho r \cot\beta + \sin(\theta - \alpha)$ is equal to 0. To rewrite the second condition, we derive an expression of θ'. We suppose again that a limb can be locally parameterized by z, and that θ is a C^1 function of z. In this case we can consider the limb equation as a function $F(\theta(z), z)$ of z only, identically equal to 0. By differentiating this function we get

$$\theta'[\rho'' \sin(\theta - \alpha) + 2\rho' \cos(\theta - \alpha) - \rho \sin(\theta - \alpha) - 2\rho\rho' r' \cot\beta] = \rho^2 r'' \cot\beta$$

By using the limb equation, we can substitute $r' \cot\beta$ and this expression simplifies into

$$\theta'[\rho\rho'' - \rho^2 - 2\rho'^2]\sin(\theta - \alpha) = \rho^3 r'' \cot\beta$$

As $[\rho\rho'' - \rho^2 - 2\rho'^2]\sin(\theta - \alpha) \neq 0$ (see section 4) we get

$$\theta'(z) = \frac{\rho^3 r'' \cot\beta}{[\rho\rho'' - \rho^2 - 2\rho'^2]\sin(\theta - \alpha)} \qquad (7)$$

Substituting θ' in the expression of C we then obtain

$$C = \frac{\cot^2\beta}{[\rho\rho'' - \rho^2 - 2\rho'^2]\sin(\theta - \alpha)} \times [\rho^4 r r'' + (\rho\rho'' - \rho^2 - 2\rho'^2)\sin^2(\theta - \alpha)\tan^2\beta]$$

So $C = 0$ iff the limb point is a cusp (Equation 6), and the lemma is proved. Note that for non transparent objects, cusps are terminations of the limbs, so the extrema of distance correspond only to extrema of r.

6. The curvature lemma

We have related the distance between the contours and the axis to the sweeping rule. We now prove the following lemma, which relates the curvature of the contours to the curvature of the sweeping rule curve (Figure 12)

Curvature lemma: *In orthographic projection, the curvature of a contour at a point is 0 iff the curvature of the sweeping rule curve is 0 at the corresponding z. The curvature is undefined at cusps.*

We first show the result for edges. An edge point can be written

$$\overrightarrow{OP}(z) = \rho(\theta_0)r(z)\vec{a} + z\vec{k}$$

Where $\vec{a}$ is the unit vector

$$\vec{a} = \cos\theta_0\vec{\imath} + \sin\theta_0\vec{\jmath}$$

The edge is a planar curve. In the $(\vec{k}, \vec{a})$ plane, its equation is $y = \rho_0 r(z)$, where $\rho_0 = \rho(\theta_0)$. This curve is simply a scaled version of the sweeping rule curve, and therefore has the same zeros of curvature. Moreover, it is known that zeros of curvature of planar curves are perspective invariant (see [8] or [17]). As the edges don't have cusps, this proves the lemma for edge projections.

To prove the lemma for limbs, we now calculate the curvature of the image of a limb of a straight homogeneous generalized cylinder in orthographic projection.

Consider a planar parameterized curve $\Gamma = (x(t), y(t))$. The curvature $\kappa(t)$ of this curve is given by

$$\kappa(t) = \frac{x''y' - y''x'}{[x'^2 + y'^2]^{3/2}}(t) = \frac{N}{D^{3/2}}(t)$$

Here the parameter t is equal to z. We omit it in the sequel for the sake of conciseness. To calculate the curvature of the limb image, we calculate x', y', x'' and y''. We have

$$x' = (\theta'\rho' r + \rho r')\sin(\theta - \alpha) + \theta'\rho r \cos(\theta - \alpha)$$
$$y' = \sin\beta - \cos\beta[(\theta'\rho' r + \rho r')\cos(\theta - \alpha) - \theta'\rho r \sin(\theta - \alpha)]$$

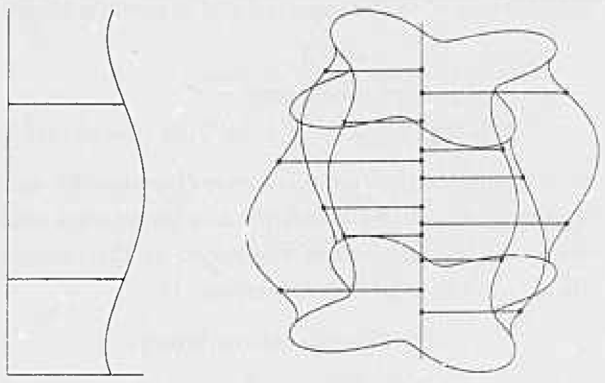


Figure 12. Zeros of curvature on the sweeping rule and the axis.

So, if we define A and B by

$$A = \theta'\rho' r + \rho r'; B = \theta'\rho r$$

We have

$$x' = A\sin(\theta-\alpha) + B\cos(\theta-\alpha)$$
$$y' = \sin\beta - \cos\beta[A\cos(\theta-\alpha) - B\sin(\theta-\alpha)]$$

We now calculate the second derivatives. We have

$$x'' = (A' - \theta'B)\sin(\theta-\alpha) + (B' + \theta'A)\cos(\theta-\alpha)$$
$$y'' = -\cos\beta[(A'-\theta'B)\cos(\theta-\alpha) - (B'+\theta'A)\sin(\theta-\alpha)]$$

With

$$A' = \theta''\rho' r + \theta'^2\rho'' r + 2\theta'\rho' r' + \rho r''$$
$$B' = \theta''\rho r + \theta'^2\rho' r + \theta'\rho r'$$

We can now rewrite $N(z) = x''y' - y''x'$

$$N(z) = \sin\beta[(A'-\theta'B)\sin(\theta-\alpha) + (B'+\theta'A)\cos(\theta-\alpha)] - \cos\beta[A(B'+\theta'A) - B(A'-\theta'B)]$$

We have

$$B' + \theta'A = \theta''\rho r + 2\theta'^2\rho' r + 2\theta'\rho r'$$
$$A' - \theta'B = \theta''\rho' r + \theta'^2\rho'' r - \theta'^2\rho r + 2\theta'\rho' r' + \rho r''$$

So

$$A(B'+\theta'A) - B(A'-\theta'B) = \\ = \theta''\rho^2 r r' - \theta'^3 r^2[\rho\rho'' - \rho^2 - 2\rho'^2] \\ + 2\theta'^2\rho\rho' r r' + \theta'\rho^2[2r'^2 - rr'']$$

We have

$$(A'-\theta'B)\sin(\theta-\alpha) + (B'+\theta'A)\cos(\theta-\alpha) = \\ = [\theta'' r + 2\theta' r' + 2\theta'^2\frac{\rho'}{\rho}r][\rho'\sin(\theta-\alpha) + \rho\cos(\theta-\alpha)] \\ + \frac{\theta' r}{\rho}[\theta'(\rho\rho'' - \rho^2 - 2\rho'^2)\sin(\theta-\alpha)] + \rho r''\sin(\theta-\alpha)$$

Using the limb equation (4) and the expression (7) of θ' this simplifies into

$$(A'-\theta'B)\sin(\theta-\alpha) + (B'+\theta'A)\cos(\theta-\alpha) = \\ = \cot\beta[\theta''\rho^2 r r' + 2\theta'\rho^2 r'^2 + 2\theta'^2\rho\rho' r r' + \theta'\rho^2 r r''] \\ + \rho r''\sin(\theta-\alpha)$$

We can now substitute in $N(z)$ and we get

$$N(z) = 2\cos\beta\theta'\rho^2 r r'' + \sin\beta\rho r''\sin(\theta-\alpha) \\ + \cos\beta\theta'^3 r^2[\rho\rho'' - \rho^2 - 2\rho'^2]$$

We suppose as before $\sin(\theta-\alpha) \neq 0$ we get by substituting $\theta'[\rho\rho'' - \rho^2 - 2\rho'^2]$

$$N(z) = \frac{\rho r''}{\sin\beta\sin(\theta-\alpha)} \\ \times [\sin^2\beta\sin^2(\theta-\alpha) + 2\sin\beta\cos\beta\theta'\rho r \\ + \cos^2\beta\theta'^2\rho^2 r^2]$$

This finally simplifies into

$$N(z) = \frac{\rho r''\sin\beta C^2}{\sin(\theta-\alpha)}$$

Where, as before

$$C = \theta'\rho r\cot\beta + \sin(\theta-\alpha)$$

We now calculate $D(z)$. We have seen in the previous section that

$$x' = \rho r' C$$

We have, if we assume as before that $\sin(\theta-\alpha) \neq 0$

$$y' = \sin\beta - \cos\beta[A\cos(\theta-\alpha) - B\sin(\theta-\alpha)] \\ = \sin\beta - \frac{\cos\beta}{\sin(\theta-\alpha)}[A\cos(\theta-\alpha)\sin(\theta-\alpha) \\ - B\sin^2(\theta-\alpha)] \\ = \sin\beta - \frac{\cos\beta}{\sin(\theta-\alpha)}[\cos(\theta-\alpha)(A\sin(\theta-\alpha) \\ + B\cos(\theta-\alpha)) - B]$$

So finally, as $A\sin(\theta-\alpha) + B\cos(\theta-\alpha) = x' = \rho r' C$

$$y' = \frac{C}{\sin(\theta-\alpha)}[\sin\beta - \rho r'\cos\beta\cos(\theta-\alpha)]$$

Let us denote $[\sin\beta - \rho r'\cos\beta\cos(\theta-\alpha)]$ by E, we have

$$y' = \frac{C}{\sin(\theta-\alpha)}E$$

and

$$E = \sin\beta[1 - \frac{\cos(\theta-\alpha)}{\rho}\rho^2 r'\cot\beta]$$

Using the limb equation (4) we get

$$E = \sin\beta[1 - \frac{\cos(\theta-\alpha)}{\rho}(\rho'\sin(\theta-\alpha) + \rho\cos(\theta-\alpha))]$$

$$= \sin\beta[\sin^2(\theta-\alpha) - \frac{\rho'}{\rho}\sin(\theta-\alpha)\cos(\theta-\alpha)]$$

So finally we have

$$E = \frac{1}{\rho}\sin\beta\sin(\theta-\alpha)[\rho\sin(\theta-\alpha) - \rho'\cos(\theta-\alpha)]$$

We can calculate E^2

$$E^2 = \frac{1}{\rho^2}\sin^2\beta\sin^2(\theta-\alpha)$$
$$[\rho^2 + \rho'^2 - (\rho'\sin(\theta-\alpha) + \rho\cos(\theta-\alpha))^2]$$

We use again the limb equation (4) to finally simplify E^2 into

$$E^2 = \frac{1}{\rho^2}\sin^2(\theta-\alpha)[\sin^2\beta(\rho^2+\rho'^2) - \cos^2\beta\rho^4 r'^2]$$

We can now substitute the values of x'^2 and y'^2 in $D(z)$. We get

$$D(z) = C^2[\rho^2 r'^2 + \frac{\sin^2\beta}{\rho^2}(\rho^2+\rho'^2) - \rho^2 r'^2\cos^2\beta]$$

So finally

$$D(z) = C^2\sin^2\beta[1 + \rho^2 r'^2 + \frac{\rho'^2}{\rho^2}]$$

We get

$$\kappa = \frac{N}{D^{3/2}} = \frac{1}{\gamma}K$$

Where

$$\gamma = C\sin^2\beta\sin(\theta-\alpha)$$

and

$$K = \frac{\rho r''}{[1 + \rho^2 r'^2 + \frac{\rho'^2}{\rho^2}]^{3/2}}$$

K has the dimensions of a curvature. In fact, if the SHGC is a solid of revolution, it is exactly equal to the curvature of the sweeping rule. In any case, $K = 0$ iff $r'' = 0$ i.e. iff the curvature of the sweeping rule curve is 0. As before, $\gamma = 0$ iff the point is a cusp (Equation 6), and the lemma is proved.

7. Finding the axis of SHGC's

We now show how we can use the intersecting tangents and curvature lemmas to find the axis of a straight homogeneous generalized cylinder in an image. Both algorithms are very simple and we don't claim they are very robust. They merely demonstrate that it is possible to use our results to segment real images containing SHGC's.

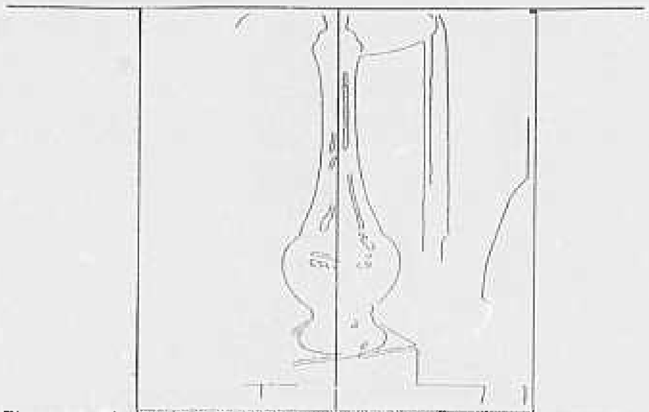


Figure 13. The axis of a lamp, found by the first algorithm of Section 7.

The first algorithm is a simple Hough transform algorithm. It is divided in five steps: find the contours in the image through edge detection; for each edgel, draw the tangent line; compute the intersections; for each intersection, find all the straight lines going through it; and finally use a Hough transform to find the actual axis.

The edge detection is done using an implementation of Canny's [5] edge detector. The algorithm is computationally expensive, as all non parallel tangents intersect, and each of the intersections generates a curve (all lines going through that point) in the Hough space. Figure 13 shows an example of the application of this algorithm to finding the axis of a lamp.

The second algorithm uses zeros of curvature to improve the efficiency of the search. It is divided into three steps: find the contours; mark all zeros of curvature, search for the matching pairs of zeros of curvature on two contours which minimize a least square error fit between the tangents' intersections and a straight line. Figure 14 shows the result of this algorithm on an other example.

Conclusion

We have derived the limb equation for generalized cylinders with straight and curved spines. We have solved it for a large class of generalized cylinders. We also have used the limb equation to prove invariant properties of the projection of generalized cylinders in images. We have used these properties to find the axis of SHGC's in real images. Our results have been restricted to the orthographic projection of generalized cylinders whose star shaped cross sections are orthogonal to their axes. It is in fact possible to extend most of our results to perspective projection, non star shaped cross sections, and oblique generalized cylinders (see [12],[14]).

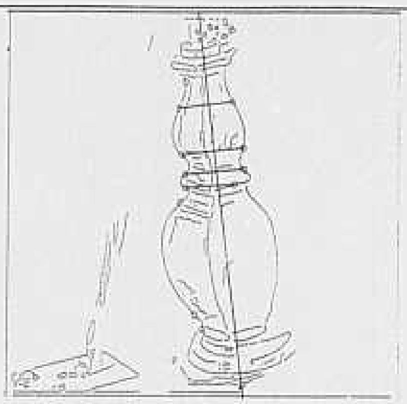


Figure 14. The axis of a lamp, found by the second algorithm of Section 7.

The results presented here are steps toward the development of a new intelligent vision system called *Successor*. We are in the process of including them in several modules of Successor. We use the solved limb equations (section 2) in the geometric modelling module [13] to display efficiently the primitives. The invariant properties (sections 3,4,5,6) are used to predict observable features in the images of the modelled primitives. The axis finding algorithms (section 7) are used in the segmentation module.

Our future work will be dedicated to three main themes. First we will generalize our results to an even broader class of generalized cylinders (e.g. with a rotating sweeping rule). Second we will develop more efficient and robust algorithms for the segmentation of images, and develop analogous methods for depth maps. Finally, we will develop matching techniques for recognizing objects represented by part-whole graphs of joined primitives [2].

References

[1] Binford,T.O.,"Visual perception by computer",Proc. IEEE Conf. on Systems and Control, Miami, December, 1971.

[2] Binford, T.O., "Inferring surfaces from images", Artificial Intelligence 17, pp. 205-244, 1981.

[3] Brady, J.M, and Asada, H., "Smooth local symmetries and their implementation", Proceedings of the First International Symposium on Robotics Research, Bretton Woods, Brady J.M., and Paul R.,(eds.), MIT Press, 1984.

[4] Brooks, R.A., "Symbolics reasoning among 3D models and 2D images", Artificial Intelligence 17, pp. 285-348, 1981.

[5] Canny, J.F., "A computational approach to edge detection", IEEE Trans. on Pattern Analysis and Machine Intelligence, PAMI-8, 6, November 1986.

[6] Koenderink, J.J., and Van Doorn, A.J., "The singularities of the visual mapping", Biological Cybernetics 24, pp. 51-59, 1976.

[7] Malik, J., "Labelling line drawings of curved objects", Proc. IU Workshop, pp. 209-218, 1985.

[8] Marimont,D.H.,"A representation for image curves", Proc. AAAI-84, Austin, 1984.

[9] Marr, D., "Analysis of occluding contour", Proc. Royal Society of London, B-197, pp. 441-475, 1977.

[10] Marr, D., and Nishihara, K., "Representation and recognition of the spatial orginazition of three dimensional shapes", Proc. Royal Society of London, B-200, pp. 269-294, 1977.

[11] Nevatia, R., "Machine perception", Prentice-Hall, Englewood Cliffs, NJ, 1982.

[12] Ponce, J., and Chelberg, D., "Finding the limbs and cusps of generalized cylinders", Proc. of the 4th IEEE Robotics Conference, 1987.

[13] Ponce, J., and Chelberg, D., "Localized intersections computation for solid modelling with straight homogeneous generalized cylinders", Proc. of the 4th IEEE Robotics Conference, 1987.

[14] Ponce, J., Chelberg, D., and Mann, W., "Invariant properties of the projections of straight homogeneons generalized cylinders", submitted to the first International Conference on Computer vision, 1987.

[15] Shafer, S.A., and Kanade, T., "The theory of straight homogeneous generalized cylinders", Carnegie Mellon Univ. Comp. Science Dept. Tech. Rep. CMU-CS083-105, 1983.

[16] Scott, R , "Graphics and prediction from models", Proc. IU Workshop, pp. 98-106, 1984.

[17] Verri, A., and Yuille, A., "Perspective projection invariants", MIT AI Lab. Memo AIM-832, 1986.

[18] Williamson, R.E., and Trotter, H.F., "Multivariable Mathematics", Prentice-Hall Inc., 1979.

SURFACE SEGMENTATION AND DESCRIPTION FROM CURVATURE FEATURES [1]

T.J. Fan, G. Medioni, and R. Nevatia
Institute for Robotics and Intelligent Systems
Departments of Electrical Engineering and Computer Science
University of Southern California
Los Angeles, California 90089-0273

ABSTRACT

This paper is a continuation of our effort [6,7,9] in which we present a method to segment and describe visible surfaces of 3-D objects. We start by extracting distinguished points which will comprise the edges of segmented surface patches, using the zero-crossings and extrema of curvature along a given direction. In our implementation we use two different methods: if the sensor provides relatively noise-free range images, we compute the principal curvatures at only one resolution, otherwise, we use a multiple scale approach and compute curvature in 4 directions 45° apart to facilitate inter-scale tracking. We then group these points into curves and classify these curves into different classes which correspond to significant physical properties such as jump boundaries, folds, and ridge lines (or smooth extrema). We then use jump boundaries and folds to segment the surfaces into surface patches and fit a simple surface to each patch to reconstruct the original objects. We believe that these descriptions not only make explicit most of the salient properties present in the original input, but are more suited to further processing, such as matching with a given model. The generality and robustness of this approach is illustrated on scene images from different available range sensors.

1 Introduction

We are interested in the description of 3-D surfaces and objects. Good descriptions of surfaces are needed for many tasks such as surface inspection, object recognition and mechanical manipulation of the objects. Surface descriptions are also important for computer graphics, though the criteria for good representation are different for this task. In this work, we assume that range data (i.e. the 3-D positions) of the points on the visible surface are available, say by the use of a laser range finder. It is also assumed that this data is *dense*, in the sense of being sampled on a certain grid and not just at discontinuities (as may be the case for uninterpolated edge-based stereo data).

To generate useful descriptions, a useful representation is needed. In general, such a description should be suitable for the task of object recognition and position identification. It should be *rich*, so that similar objects can be identified, *stable*, so that local changes do not radically alter the descriptions, and have *local support* so that partially visible objects can be identified. It should also enable us to recreate, from its features, a shape reasonably close to the original one. These criteria naturally lead us to segmented, hierarchical descriptions.

Here, our interest is primarily in the description of surfaces, but we also believe that surface descriptions may be an important tool in generating generalized cylinder descriptions [11]. It is our thesis that the methodology of segmented, hierarchical descriptions that has proved useful for generalized cylinder representations can also be usefully applied for surface descriptions. The key is to find the natural segmentations of a surface, rather than those forced by an arbitrary approximating scheme.

In particular, we propose that the following surface points and lines are critical for a natural segmentation of the surface:

1. **jump boundaries** where the surface undergoes a discontinuity
2. **folds** (also called **creases**) which correspond to surface orientation discontinuities
3. **ridge lines** which correspond to smooth local extrema of curvature

We show that one way of inferring these significant surface features is by examining the *zero-crossings* and *extremal* values of surface curvature measures. We then use the detected features, and their descriptions, to segment a complex surface into simpler, meaningful components. These *patches* can then be approximated by simple surface models, if desired. The next step would be a grouping of the patches to form meaningful 3-D objects, we have not yet implemented this step.

Some other authors have also used curvature properties to achieve surface segmentation [2,5]. Our approach is similar to that of Ponce and Brady [10] but differs in detail and, we believe, goes much farther towards a complete surface description.

2 Detection of Surface Features Using Curvature Properties

2.1 Overview

Our approach is to develop methods for finding surface discontinuities and other features described in section 1 explicitly. Fur-

[1] This research was supported by the Defense Advanced Research Projects Agency under contract number F33615-84-K-1404, monitored by the Air Force Wright Aeronautical Laboratories, Darpa Order No. 3119.

ther, we propose that computing curvature properties of the surface is useful in finding these features. We first segment the scene into patches and then approximate each patch, which is known to be simple.

It is well known in differential geometry that a surface can be reconstructed up to second order from the knowledge of curvature at each point, except for a constant term (by using the first and second fundamental forms, see [8] for example). The curvature of a surface at a given point varies with the direction in which it is measured. A differential geometry theorem [8], however, tells us that from the curvature in two distinguished orthogonal directions, known as the *principal directions*, it is possible to compute curvature in any direction at that point. We also show, in appendix, that the same result can be achieved by computing curvature in 4 different directions, 45° apart. However, our main goal is not exact reconstruction of the surface but location and description of significant features. We show that *zero-crossings* and *extrema* of curvature in chosen directions are useful for this as explained below.

We find that an occluding boundary (sometimes referred to as a **jump boundary** in range data analysis) creates a zero-crossing of the curvature in a direction normal to that of the boundary. A **fold boundary** (where surface normals are discontinuous) causes a local extremum of the curvature at that point. Fold boundaries may also create zero-crossings away from the location of the boundary itself. Lastly, **curvature extrema** correspond to certain distinguished points or lines on smooth surfaces, such as along the extrema of the major axis of the cross-section of an elliptical cylinder. We illustrate these properties by some examples here, a more analytical treatment can be found in [10].

2.2 Detailed description

The properties of curvature as they relate to the input are rather simple for one-dimensional signals. Extension to the 2-dimensional case requires some caution: It seems natural to compute at every point the two principal curvatures and to concentrate on the behavior of the one with the largest magnitude. If multiple scales are used, however, not only the location, but also the orientation of features of interest change, making the tracking of these features most difficult.

To get around this tracking problem, we have developed two strategies:

- If the range images are noisy or if texture is present, then we use a multiple scale approach. Instead of computing principal curvatures, however, we compute at each point directional curvatures in 4 different directions 45° apart (It is equivalent to computing the two principal curvatures, as proved in the appendix). From these, we compute zero-crossings and extrema in the 4 directions. The advantage, of course, is the tracking becomes easier as it is performed along a single dimension, and the disadvantage being that a merging phase is necessary. This method will be referred to as method 1.
- If the range images are relatively noise-free and shape exhibits itself at one level (as is very common in most range images that we have scanned), then we use the straightforward implementation in which we compute first the two principal curvatures, then the zero-crossings and extrema of the largest principal curvature. This method will be referred to as method 2.

The block diagrams for Method 1 and 2 are shown in Figure 1 and Figure 2, respectively. A block diagram of the complete approach is given in Figure 3. The detailed description of each step can be found in [6]. To illustrate the steps of our two methods, we will use the example of a *cup* with an elliptical cross-section shown in Figure 4. This data was obtained using an active stereo range finding system at INRIA [4] (courtesy of Dr. Fabrice Clara). The elliptical effect was created artificially by scaling the data. The resolution of the data is 80 x 100 pixels.

Figure 5 shows the results after scale-space tracking in vertical direction for the "cup" image using method 1.

Figure 6 shows the features detected for "cup" image using method 2.

Either of these two methods produces a set of pointwise descriptions. During the next step, we group and label them as:

- isolated positive or negative extrema (+ or −)
  these correspond to **folds** if they are steep, or to **smooth curvature extrema** if they are smooth.
- extrema linked to one zero-crossing (+ 0 or − 0)
  these correspond to **folds**.
- zero-crossing flanked by 2 extrema (+ 0 −)
  these correspond to **jump boundaries**.

Finally, we link these labels into curves.

Figure 7 shows the descriptors on the cup data where Fig. 7(a) shows jump boundaries, Fig. 7(b) shows folds, Fig. 7(c) shows positive curvature extrema, Fig. 7(d) shows negative curvature extrema, Fig. 7(e) combines jump boundaries and folds.

3 Surface Segmentation, Description and Reconstruction

The features detected from the processes described above are central to our approach to segmenting a given surface. These features provide us with *partial* boundaries for patches in which the surface should be segmented but not necessarily a *complete* segmentation. We start with boundaries that correspond to jump boundaries and folds, we close these boundaries by extending curves as described below. The resulting *regions* are assumed to correspond to elementary surface patches. These regions may have to be further segmented, either based on the region shape or on the results of surface fitting. In the latter case, the smooth extrema that we have found may help us in defining new boundaries for further segmentation (we have not implemented this step).

3.1 Forming Closed Regions

We do not expect our feature detection to perform flawlessly, especially in busy areas where more than two different surface patches meet, as the model does not account for such transitions. This can be seen in Figure 7 at the locations where the handle and the body of the cup merge. It is important to separate such regions, however, since it is impossible to fit a single surface patch over two different regions.

We handle these situations by noting that they occur only when features are detected very close to each other (zero-crossings cannot be closer than 2σ, where σ is the variance of the LoG mask). So, for each pixel in a given region, we define its *radius* as the radius of the largest disk enclosing this pixel without including any boundary point. We mark each pixel with radius smaller than a fixed threshold (we choose 3 here) and then consider all marked points with at least one unmarked neighbor as new boundary points. The new patches created by these boundaries are kept only if they are large enough (currently 1% of the largest region). This simple technique proved to be quite powerful on our database of range images. We have used the above descriptors for the cup and extracted contours of surface patches as shown in Figure 8.

3.2 Surface Fitting

Once we have obtained closed boundaries, it becomes easy to describe the surface properties of each patch: We approximate each patch by a bivariate polynomial, here simply of second order (biquadratic) for valid reasons: Plane fitting would create large errors and is not appropriate outside of the blocks world, and third or higher order polynomials must be avoided as they might introduce oscillations which, if actually present, would have been detected by our feature extraction process. The approximating function is therefore

$$g(x,y) = a_{00} + a_{10}x + a_{01}y + a_{20}x^2 + a_{11}xy + a_{02}y^2$$

The coefficients are obtained by minimizing the least-squares error between actual and interpolated data. We have found that it is not necessary to use all the points in a patch to obtain a good approximation, but that a band of values close to the detected contours, and, of course, the points of the patch detected as smooth extrema suffice. This is illustrated on Figure 9 for the "cup" image, where the left column displays the original range images from different view points, and the right column shows the reconstructed range images from the corresponding view points. We can see from the results that all patches are very well approximated, and the noise in the original range image has been smoothed out.

It is important to note, however, that there exist simple surfaces which cannot be well represented by such a polynomial, such as the visible parts of a sphere or a torus. Furthermore, it is impossible to extract physical edges within these patches to obtain smaller patches and reduce the error due to the fit. Depending on the applications, different approaches may be used to resolve this type of problem: If the goal is indeed accurate approximation, then we could subdivide the patch arbitrarily and approximate each subpatch so that adjacent pieces are smoothly joined, and B-splines are a good candidate. Other possibilities are to use higher-level knowledge if available.

What we have achieved so far is segmented surface description, not object description, but we believe it is a large step in that direction.

3.3 Further Descriptions

Labels and surface patches can be combined to give a very good description for 3-D surfaces. The relationship between surface patches can be established from the knowledge of the label type at each patch boundary. Furthermore, from the direction of the change of curvature signs (from negative to positive, for example) around the zero-crossings, we can tell occluding surfaces from occluded surfaces. Figure 10 shows the regions and surrounding boundaries and Figure 11 shows the relationships between regions.

Using labels, surface patches information and relationships between patches, we aim at obtaining *object* description, or rich graph descriptions, which we believe can be used for the tasks of recognition or identification.

4 Results and Conclusions

Several other examples are shown (method 2 is chosen for feature detection) on range images from different sensors: Figure 12 is a synthetic image at low resolution (60x60) to show the effects of severe quantization noise. Figure 13 and 14 are range images obtained from ERIM (courtesy of Prof. R. Jain). The sensor is a time of flight range finder and the images are 128x128, coded on 8 bits. Figure 15 is obtained from the INRIA range finder, courtesy of Dr. J. Ponce. The image is 128x128, on 8 bits.

For each of these figures,

- (a) shows the original range images represented by intensity.
- (b) shows the folds,
- (c) shows the jump boundaries,
- (d) shows the combination of jump boundaries and folds,
- (e) shows the surface region boundaries which are computed from jump boundaries and folds, and
- (f) shows the reconstructed range images, respectively.

The program is written in Lisp and runs on a Symbolics Lisp Machine. The approximate CPU time for each data is summarized in Table 1.

Range Image	Size	Time in Seconds			
		FD	SS	SR	Total
cup	80x100	47	50	43	140
bottle	60x60	25	36	39	100
coffee cup	128x128	90	135	140	365
two cylinders	128x128	75	130	115	320
crod	128x128	105	120	160	385

FD:	Feature Detection and Description
SS:	Surface Segmentation
SR:	Surface Fitting and Reconstruction

Table 1: CPU time for each data.

We can draw some conclusions from these examples. First, significant occluding boundaries and fold boundaries are detected well (though jump boundaries would be detected as well or better by any normal edge detector also). In addition, other significant curves are also detected on the cup, the line in the middle corresponds to the apex of the elliptical cross-section. In the half bottle of Figure 12, we detect the curvature extrema where the bottle cross-section is the largest and also where it is the smallest.

We would have liked to present a detailed comparison of our method with others aimed at similar goals, particularly with works of Brady and Ponce [10] and Besl and Jain [1,3]. However, we are unable to do so for the same reasons that such comparisons are often not made in computer vision research. Both of the previous papers present results on only a small number of examples (as is common in vision papers) and algorithms are too complex for us to duplicate to test on our examples. Further, in case of Brady and Ponce paper we feel that crucial details of their use of scale-space tracking are left out. Hence, we will provide a qualitative comparison only.

We feel that our results in detecting features are similar to those obtained by Brady and Ponce; one of our methods is considerably simpler than what we understand their method to be. Our system also proceeds to compute surface segmentation and description; Brady and Ponce stop at discontinuity detection. Brady and Ponce also do not detect or use isolated extrema, which we feel are important for some types of surfaces.

Besl and Jain do provide surface segmentation and their final results on the cup example (figure 6 in [3]) are very similar to ours. However, we feel that our method is much more stable and requires use of fewer parameters. Besl and Jain start with a large number of small patches that must be merged to yield very few patches, we believe that this process could be very sensitive to a choice of merging parameters. In any case, our approach provides a complementary alternative.

We believe that our results show the essential utility and feasibility of the proposed representation scheme, though many of the details can be improved. We believe that these descriptions are useful by themselves, though we plan to group them into a yet higher level structure before using them for recognition and inspection.

Appendix

In this appendix, we show that, at every point, computing the principal curvatures (κ_1, κ_2) and their orientation α is equivalent to computing curvature in 4 different directions 45° apart $(\kappa_0, \kappa_{45}, \kappa_{90}, \kappa_{135})$. Figure 16 gives a geometric illustration for the case where κ_1 and κ_2 are positive.

Computing curvature in any direction ϕ from the principal curvatures is easy:

$$\kappa_\phi = \kappa_1 \cos^2(\phi - \alpha) + \kappa_2 \sin^2(\phi - \alpha) \tag{1}$$

Going the other way is as follows:

$$\kappa_0 = \kappa_1 \cos^2 \alpha + \kappa_2 \sin^2 \alpha \tag{2}$$

$$\kappa_{90} = \kappa_1 \sin^2 \alpha + \kappa_2 \cos^2 \alpha \tag{3}$$

$$\kappa_{45} = \kappa_1 \cos^2(\frac{\pi}{4} - \alpha) + \kappa_2 \sin^2(\frac{\pi}{4} - \alpha) \tag{4}$$

$$\kappa_{135} = \kappa_1 \sin^2(\frac{\pi}{4} - \alpha) + \kappa_2 \cos^2(\frac{\pi}{4} - \alpha) \tag{5}$$

(4) and (5) can be rewritten as:

$$2\kappa_{45} = \kappa_1 + \kappa_2 + (\kappa_1 - \kappa_2)\sin(2\alpha) \tag{6}$$

$$2\kappa_{135} = \kappa_1 + \kappa_2 + (\kappa_1 - \kappa_2)\sin(2\alpha) \tag{7}$$

Adding (2) through (5), we get

$$\kappa_1 + \kappa_2 = \frac{1}{2}(\kappa_0 + \kappa_{45} + \kappa_{90} + \kappa_{135}) \tag{8}$$

Multiplying (2) by (4) and (3) by (5) yield:

$$\begin{aligned} \kappa_0\kappa_{90} &= \kappa_1\kappa_2(\cos^4\alpha + \sin^4\alpha) + (\kappa_1^2 + \kappa_2^2)\cos^2\alpha\sin^2\alpha \\ &= \kappa_1\kappa_2 + (\kappa_1 - \kappa_2)^2\cos^2\alpha\sin^2\alpha \\ &= \kappa_1\kappa_2 + \tfrac{1}{4}(\kappa_1 - \kappa_2)^2\sin^2 2\alpha \end{aligned} \tag{9}$$

$$\begin{aligned} \kappa_{45}\kappa_{135} &= \kappa_1\kappa_2(\cos^4(\tfrac{\pi}{4} - \alpha) + \sin^4(\tfrac{\pi}{4} - \alpha)) \\ &\quad + (\kappa_1^2 + \kappa_2^2)\cos^2(\tfrac{\pi}{4} - \alpha)\sin^2(\tfrac{\pi}{4} - \alpha) \\ &= \kappa_1\kappa_2 + (\kappa_1 - \kappa_2)^2\cos^2(\tfrac{\pi}{4} - \alpha)\sin^2(\tfrac{\pi}{4} - \alpha) \\ &= \kappa_1\kappa_2 + \tfrac{1}{4}(\kappa_1 - \kappa_2)^2\sin^2(\tfrac{\pi}{2} - 2\alpha) \\ &= \kappa_1\kappa_2 + \tfrac{1}{4}\cos^2 2\alpha \end{aligned} \tag{10}$$

Adding (9) and (10) and multiplying by 4:

$$\begin{aligned} 4(\kappa_0\kappa_{90} + \kappa_{45}\kappa_{135}) &= 8\kappa_1\kappa_2 + (\kappa_1 - \kappa_2)^2 \\ &= 4\kappa_1\kappa_2 + (\kappa_1 + \kappa_2)^2 \end{aligned} \tag{11}$$

We now have $P = \kappa_1\kappa_2$ and $S = \kappa_1 + \kappa_2$, therefore κ_1 and κ_2 are solutions of the equation

$$x^2 - Sx + P = 0 \tag{12}$$

and, by convention, we have $|\kappa_1| > |\kappa_2|$, all is left is to find the value of α mod π.

Rewriting (2) as

$$\kappa_0 = \kappa_1 + (\kappa_2 - \kappa_1)\sin^2\alpha \tag{13}$$

gives us 2 solutions for α: α and $\pi - \alpha$

Rewriting (6) as

$$2\kappa_{45} = \kappa_1 + \kappa_2 + (\kappa_1 - \kappa_2)\sin 2\alpha \tag{14}$$

gives us 2 solutions for α: α and $\frac{\pi}{2} - \alpha$. The intersection of these 2 sets of equations gives us a unique solution for α mod π.

References

[1] P.J. Besl and R.C. Jain, "Intrinsic and Extrinsic Surface Characteristics", In *Proceedings of the IEEE Computer Vision and Pattern Recognition*, pages 226–233, San Francisco, California, June 9-13 1985.

[2] P.J. Besl and R.C. Jain, "Three-Dimensional Object Recognition", In *ACM Computing Surveys*, 17(1):75–145, March 1985.

[3] P.J. Besl and R.C. Jain, "Segmentation Through Symbolic Surface Descriptions", In *Proceedings of the IEEE Computer Vision and Pattern Recognition*, pages 77–85, Miami Beach, Florida, June 22-26 1986.

[4] J.D. Boissonnat and O.D. Faugeras, "Triangulation of 3-D Objects", In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, pages 658–660, Vancouver, B.C., Canada, August 24-28 1981.

[5] M. Brady, J. Ponce, A. Yuille, and H Asada, "Describing Surfaces", In H. Hanafusa and H. Inoue, editors, *Proceedings of the 2nd International Symposium on Robotics Research*, Massachusetts Institute Technology Press, Cambridge Mass., 1985.

[6] T.J. Fan, G. Medioni, and R. Nevatia, "Description of Surfaces from Range Data", In *Proceedings of DARPA Image Understanding Workshop*, pp. 232-244, Miami Beach, Florida, December 9-10, 1985.

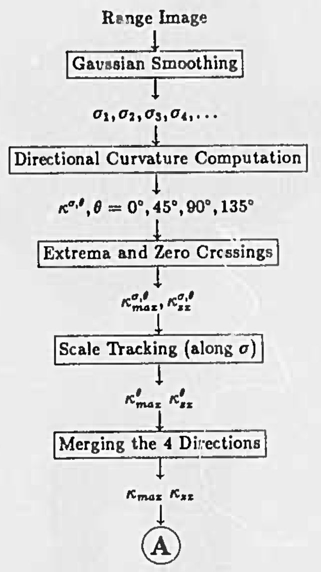
[7] T.J. Fan, G. Medioni, and R. Nevatia, "Description of Surfaces from Range Data Using Curvature Properties", In *Proceedings of the IEEE Computer Vision and Pattern Recognition*, pp. 86-91, Miami Beach, Florida, June 22-26 1986.

[8] M. Lipschutz, *Differential Geometry*, McGraw-Hill, 1969.

[9] G. Medioni and R. Nevatia, "Description of 3-D Surfaces Using Curvature Properties", In *Proceedings of DARPA Image Understanding Workshop*, October 1984.

[10] J. Ponce and M. Brady, "Toward a Surface Primal Sketch", In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 420–425, St. Louis, Mo., March 25-28 1985.

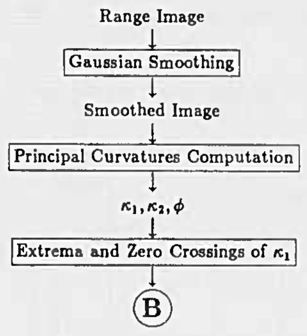
[11] K. Rao and R. Nevatia, "Generalized Cone Descriptions From Sparse 3-d Data", In *Proceedings of the IEEE Computer Vision and Pattern Recognition*, pages 256–263, Miami Beach, Florida, June 22-26 1986.



Range Image
Gaussian Smoothing
$\sigma_1, \sigma_2, \sigma_3, \sigma_4, \ldots$
Directional Curvature Computation
$\kappa^{\sigma,\theta}, \theta = 0°, 45°, 90°, 135°$
Extrema and Zero Crossings
$\kappa^{\sigma,\theta}_{max}, \kappa^{\sigma,\theta}_{zz}$
Scale Tracking (along σ)
$\kappa^{\theta}_{max}\ \kappa^{\theta}_{zz}$
Merging the 4 Directions
$\kappa_{max}\ \kappa_{zz}$
A

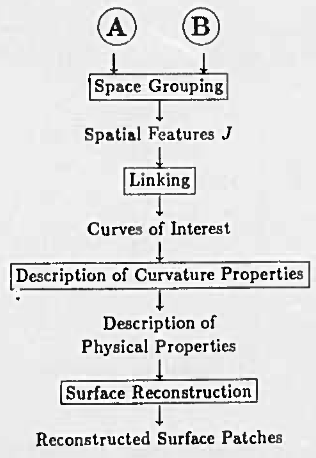
$\kappa^{\sigma,\theta}$: curvature in θ direction, after smoothing with a Gaussian mask of variance σ
κ_{max}: extrema
κ_{zz}: zero-crossing

Figure 1: Block diagram for computing significant curvature features by Method 1.



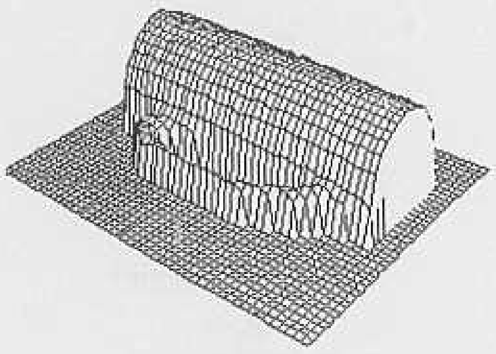
Range Image
Gaussian Smoothing
Smoothed Image
Principal Curvatures Computation
κ_1, κ_2, ϕ
Extrema and Zero Crossings of κ_1
B

Figure 2: Block diagram for computing significant curvature features by Method 2.



A
B
Space Grouping
Spatial Features J
Linking
Curves of Interest
Description of Curvature Properties
Description of Physical Properties
Surface Reconstruction
Reconstructed Surface Patches

Figure 3: Block diagram for the complete approach.


(a) 3-D plot

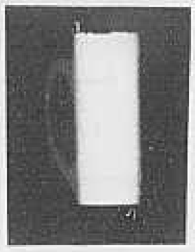
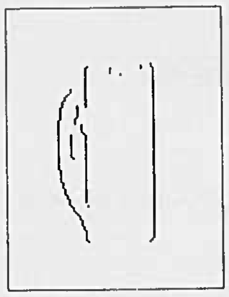
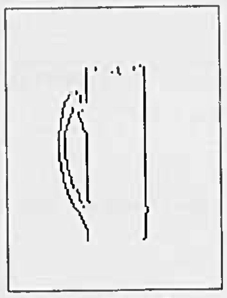

(b) Range represented by intensity

Figure 4: 3-D plot of the "cup" image and its intensity representation.


(a) Positive extrema


(b) Zero-crossings

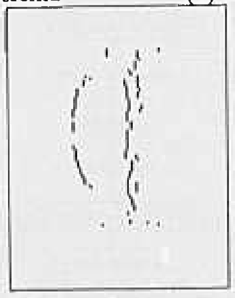
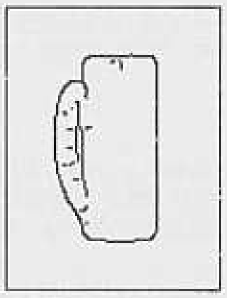
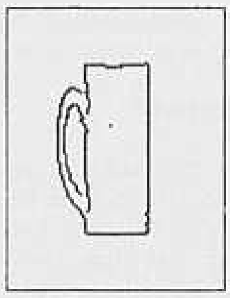

(c) Negative extrema

Figure 5: Results of the scale-space tracking in vertical direction for the "cup" image using method 1.


(a) Positive extrema


(b) Zero-crossings

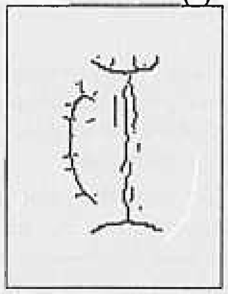
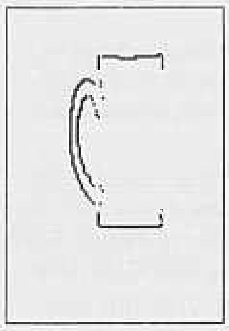
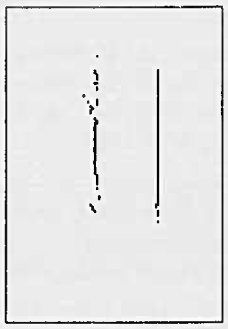
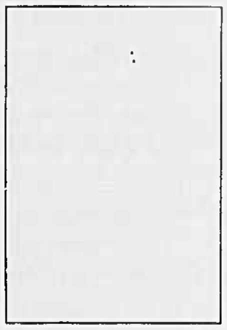
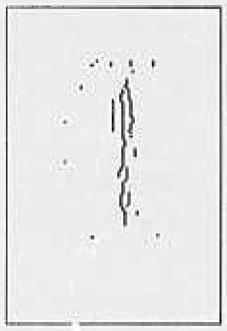

(c) Negative extrema

Figure 6: Features detected for "cup" image using method 2.


(a) Jump boundaries


(b) Folds


(c) Positive extrema


(d) Negative extrema

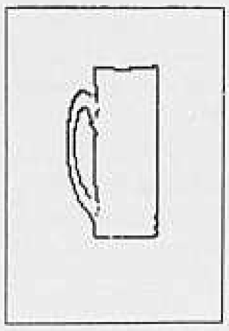

(e) Combination of (a) and (b)

Figure 7: Results of the space grouping procedure for "cup" image.

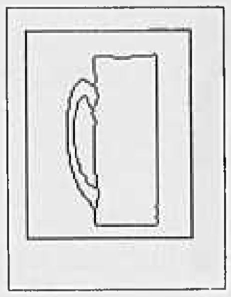


Figure 8: Closed boundaries of the "cup" image.

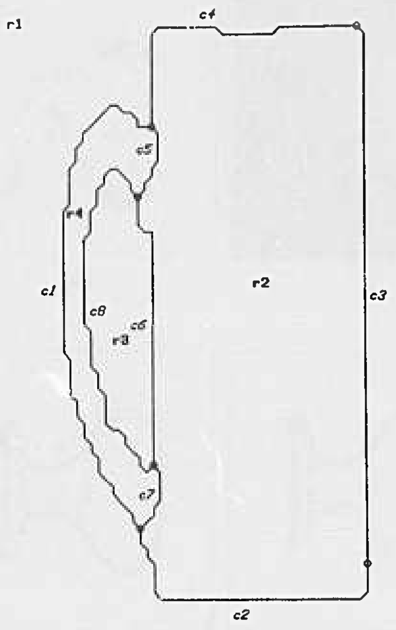


Figure 10: Regions and surrounding boundaries for the "cup" image.

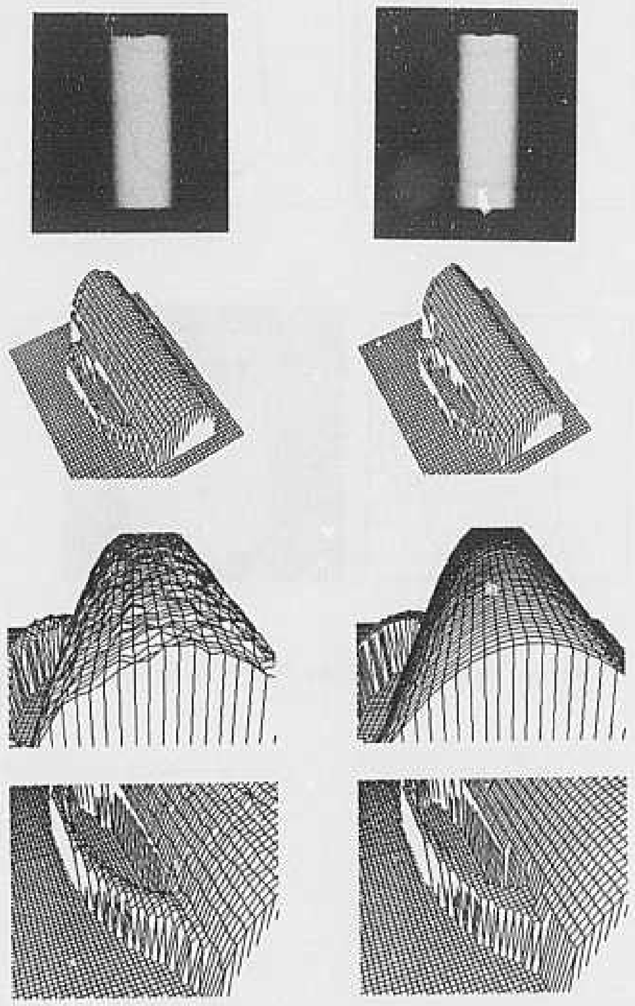


Figure 9: Reconstruction of the "cup" image.

```
r1 is PLANAR BACKGROUND, surrounded by curves c1 c2 c3 c4
   IS-OCCLUDED-BY r2 AT c2
   MAY-BE-OCCLUDED-BY r2 AT c3
   IS-OCCLUDED-BY r2 AT c4
   IS-OCCLUDED-BY r4 AT c1

r2 is CONVEX-PARABOLIC, surrounded by curves c2 c3 c4 c5 c6 c7
   OCCLUDES r1 AT c2
   MAY-OCCLUDES r1 AT c3
   OCCLUDES r1 AT c4
   MAY-OCCLUDES r3 AT c6
   OCCLUDES r4 AT c5
   OCCLUDES r4 AT c7

r3 is PLANAR BACKGROUND, surrounded by curves c6 c8
   MAY-BE-OCCLUDED-BY r2 AT c6
   IS-OCCLUDED-BY r4 AT c8

r4 is CONCAVE-PARABOLIC, surrounded by curves c1 c5 c7 c8
   OCCLUDES r1 AT c1
   IS-OCCLUDED-BY r2 AT c5
   IS-OCCLUDED-BY r2 AT c7
   OCCLUDES r3 AT c8
```

Figure 11: Relationships between regions of "cup" image.

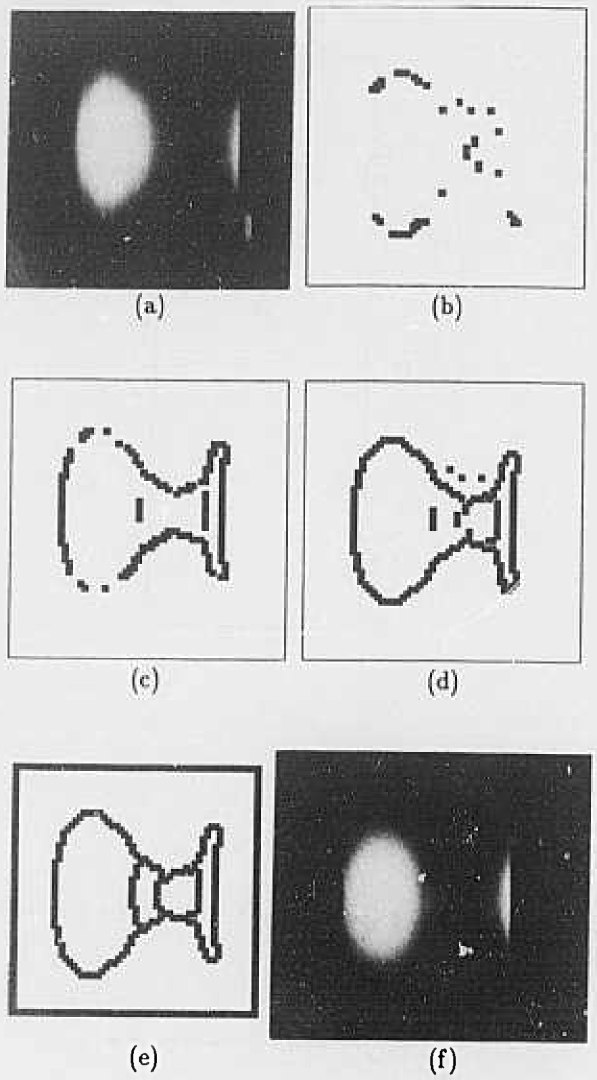


Figure 12: Results of the "bottle" image (synthetic).

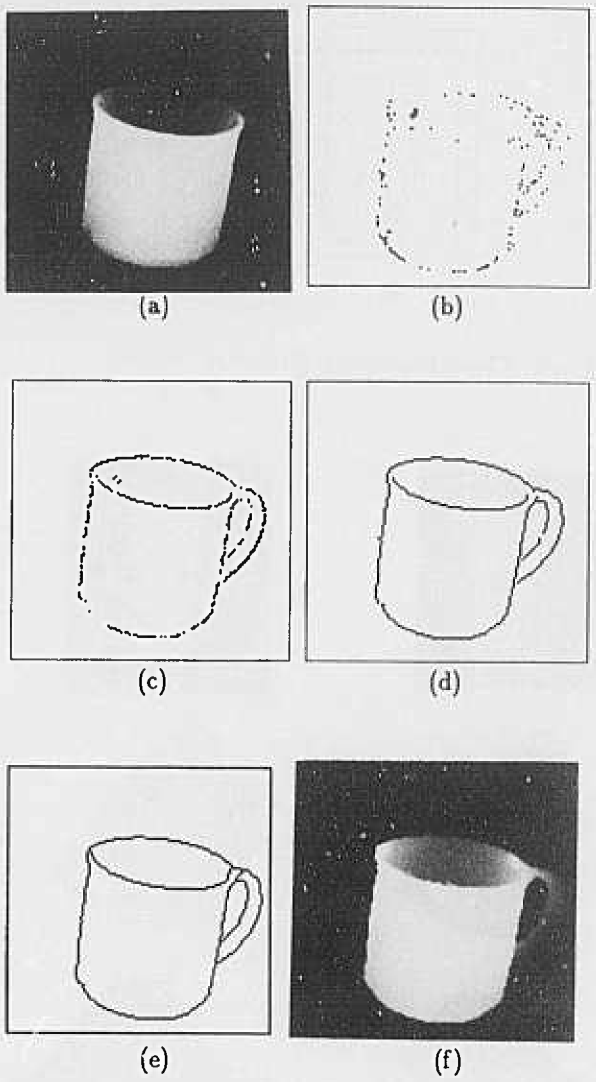


Figure 13: Results of the "coffee cup" image.

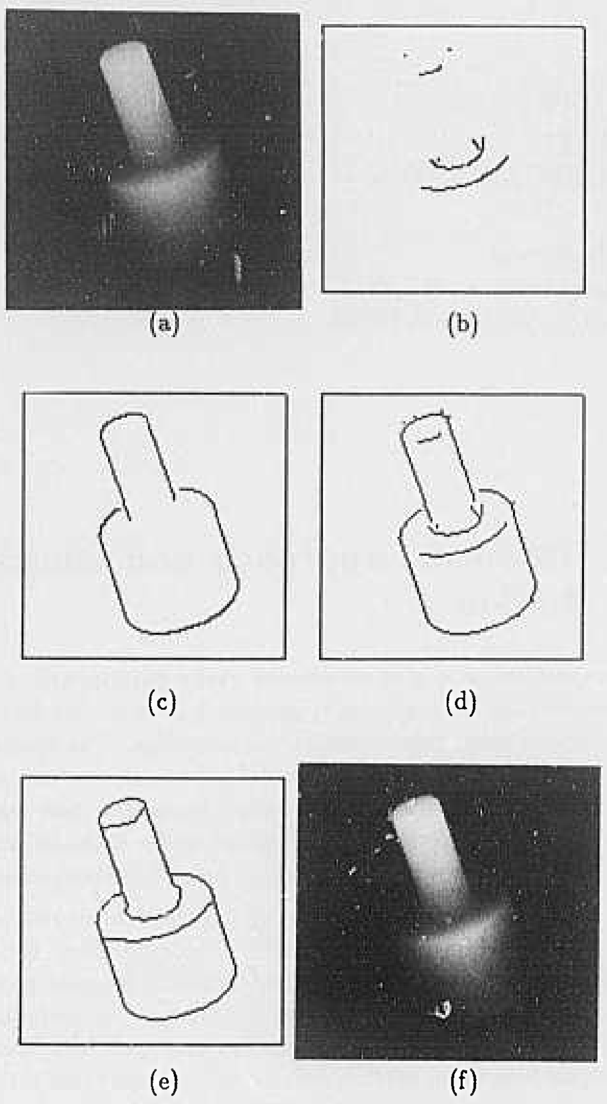


Figure 14: Results of the "two cylinders" image.

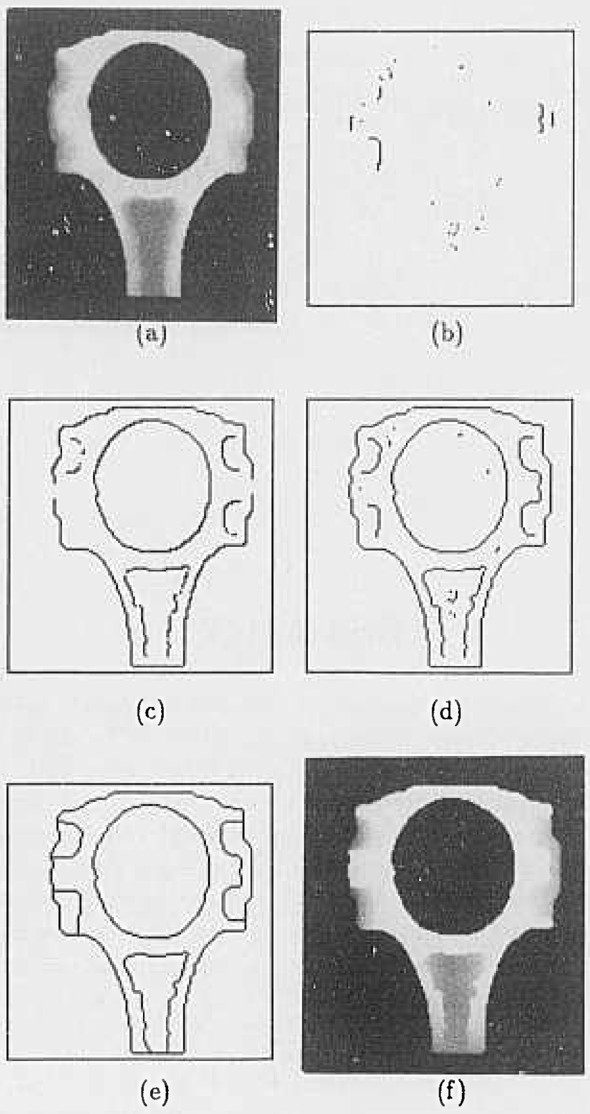


Figure 15: Results of the "crod" image.

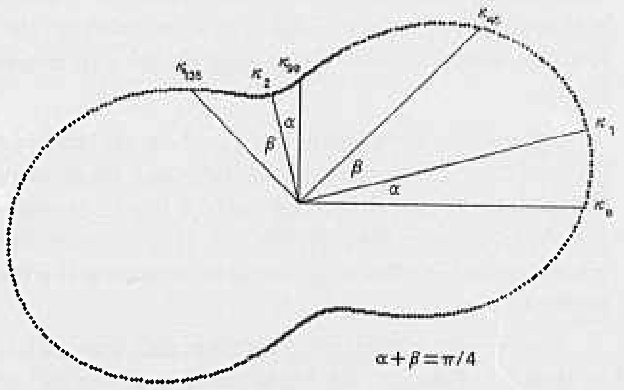


Figure 16: Relation between principal curvatures and directional curvatures.

FROM SPARSE 3-D DATA *DIRECTLY* TO VOLUMETRIC SHAPE DESCRIPTIONS[1]

Kashipati Rao and R. Nevatia,
Institute for Robotics and Intelligent Systems, MC-0273,
University of Southern California, Los Angeles, CA-90089.

ABSTRACT

We present an approach to compute volumetric shape descriptions *directly* from sparse, imperfect 3-D data with noise, surface markings and missing information such as that obtained from stereopsis or other "shape from" methods. The shape descriptions are in terms of generalized cones. In this paper, we explain the changes made to our system since our last paper [1], and then present results on real data for Linear Straight Homogeneous Generalized Cones. We conclude by briefly discussing the problem of more complex scenes.

1 Introduction

Shape description is a key problem in robotics vision. It is useful for a number of tasks like recognition, assembly, inspection, grasping and reasoning. The data available for such tasks is usually imperfect and may also be sparse. Previous approaches based on perfect line drawings or dense data will, therefore, have great difficulty, if not fail, with such data. In this paper, we focus on the problem of description and segmentation of a scene from sparse, imperfect 3-D data with surface markings, noise and missing information. We propose a new approach of going *directly* to a volume-based approach and the general model of the objects we consider is the generalized cone. We explain the changes made to our system since our last paper [1], and then present results with real data for Linear Straight Homogeneous Generalized Cones. We conclude by discussing the problem of more complex generalized cones and suggesting possible solutions.

[1]This research is supported by the Defense Advanced Research Projects Agency under contract number F33615-84-K-1404, monitored by the Air Force Wright Aeronautical Laboratories, Darpa Order No. 3119.

2 Problem, approach and contribution

Our problem is to give structured shape descriptions of a scene starting from sparse, imperfect, 3-D data. We desire a volume-based, object-centered descriptions. The sparse, imperfect, 3-D data may be available from stereo, motion or other shape-from methods. Such imperfect data may also be obtained from an active range finder if the objects in the scene have dark spots or poor reflectivity properties.

Previous approaches [2,3] using 3-D data from sources such as above, have built a surface by interpolation. However, there are problems in those schemes because good interpolation requires knowledge of the object boundaries, that is, the solution of the segmentation problem itself. Also, such an interpolation scheme will not work well if the data is not spread *throughout* the region being interpolated in. Moreover, a surface description scheme is not necessary if our ultimate objective is a volume-based, object-centered description. We have therefore adopted the *more direct* volume-based approach. These two approaches, in the perspective of vision research in general, are summarized in figure 1.

Our volume descriptions are based on *generalized cones* (GCs) [4,5]. Previous work on GCs has required dense, range data [6] and has used perfect line drawings [7,8]. Brooks [9] handles fragmented 2-D data but uses detailed knowledge of the objects and some knowledge of the viewing position.

A scene with objects describable as GCs may be labelled as shown in figure 2. By "axial contour generator" or acg (note the change in terminology from just "contour generator" used in [1]) we mean that part of the occluding boundary of the GC that is along its length. By terminators of a GC we simply mean its ends. A scene may also have surface marks and missing information as shown in figure 2.

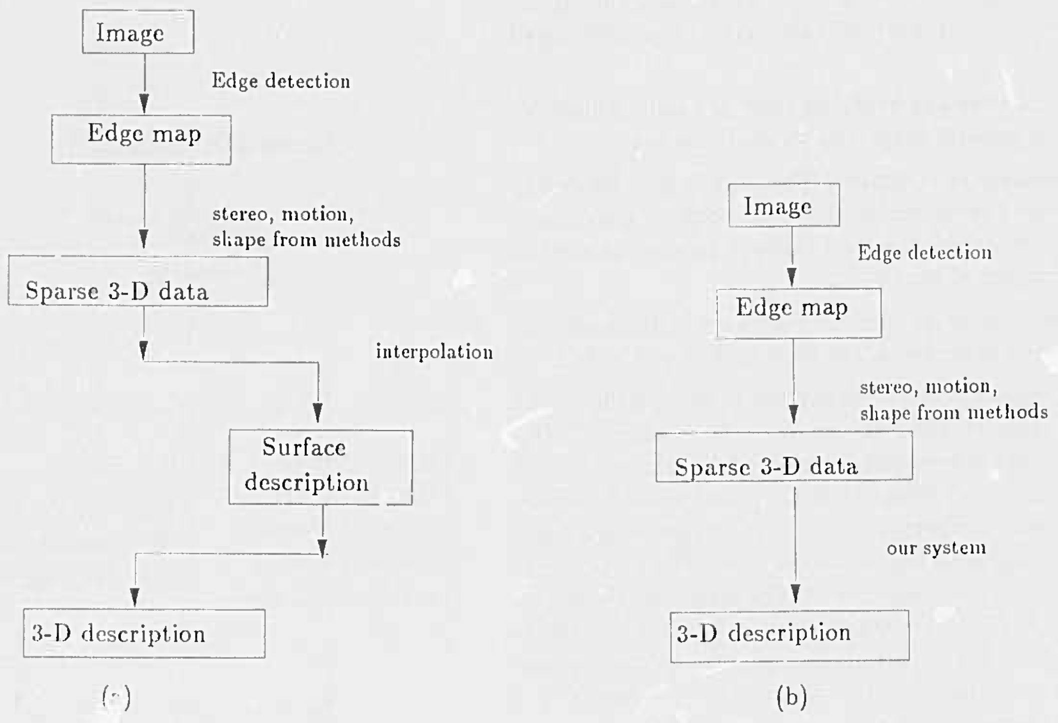


Figure 1: Two approaches in computer vision are contrasted in this figure. (a) shows one approach quite popular in vision research. (b) shows our approach, which departs from the previous approach in that it goes directly from sparse 3-D data to a volume-based description. Note, we do not go through the intermediate surface description stage.

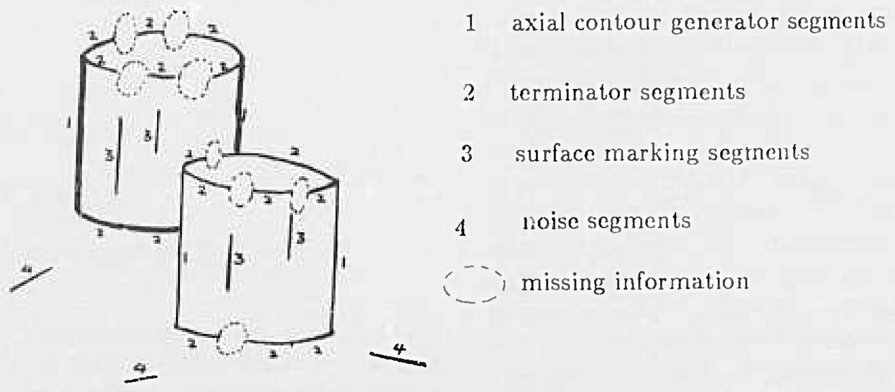

1 axial contour generator segments
2 terminator segments
3 surface marking segments
4 noise segments
missing information

Figure 2: Labelling a scene with generalized cones

The general approach in this research is based on the hypothesize and verify paradigm. That is, hypothesize a description of shape using one piece of evidence and verify using another piece. The pieces of evidence we shall use are the axial contour generators and terminators mentioned above.

To hypothesize and verify we shall use some properties of GCs. The general properties we shall use are:

In-betweenness/extremity: The terminator boundary lies completely within the axial contour generators. In other words, the axial contour generators are the extremities of the object.

Tangency: The axial contour generator is tangential to the terminator boundary both in 3-D and in 2-D.

The other properties that we use are those specific to the particular class of GCs we consider. For classes of GCs we shall use the terminology developed by Shafer [10]. In this terminology, a *homogeneous* GC has a constant cross-section shape, a *linear* GC has a linear cross-section function and a straight GC has a *straight* axis. The first class of GCs we study is the Linear Straight Homogeneous Generalized Cone (LSHGC). For these, *the axial contour generators are planar from any view* (from Shafer [10]). In the concluding section we briefly mention the specific properties of a more general class of GCs.

3 Changes to the system

In this section we shall describe the changes to the system, after giving a brief background of the old system for the sake of completeness. A detailed exposition of the system may be found in [1].

A block diagram of the system is given in figure 3. The input to the system is a sparse set of 3-D line segments, made available by lower-level programs. After preliminary processing (finding relationship between lines, finding junctions), we search for GCs. The methods used are: axial contour generator directed method and terminator directed method. The axial contour generator directed method finds a pair of candidate axial contour generators (acgs) first. These are long segments and, for LSHGCs, the pair has to be coplanar. Then the method verifies the hypothesis by finding corresponding terminators. This is done by tracing a contour between the acgs while satisfying the "in-betweenness" and tangency properties. The terminator directed method, on the other hand, finds candidate terminators first by finding maximal, coplanar, convex sets, ordering them and finding the corners of the sets. It then verifies the hypothesis of a GC by finding the corresponding acgs using the tangency and "in-betweenness" (extremity) properties and, for LSHGCs, the coplanarity property too.

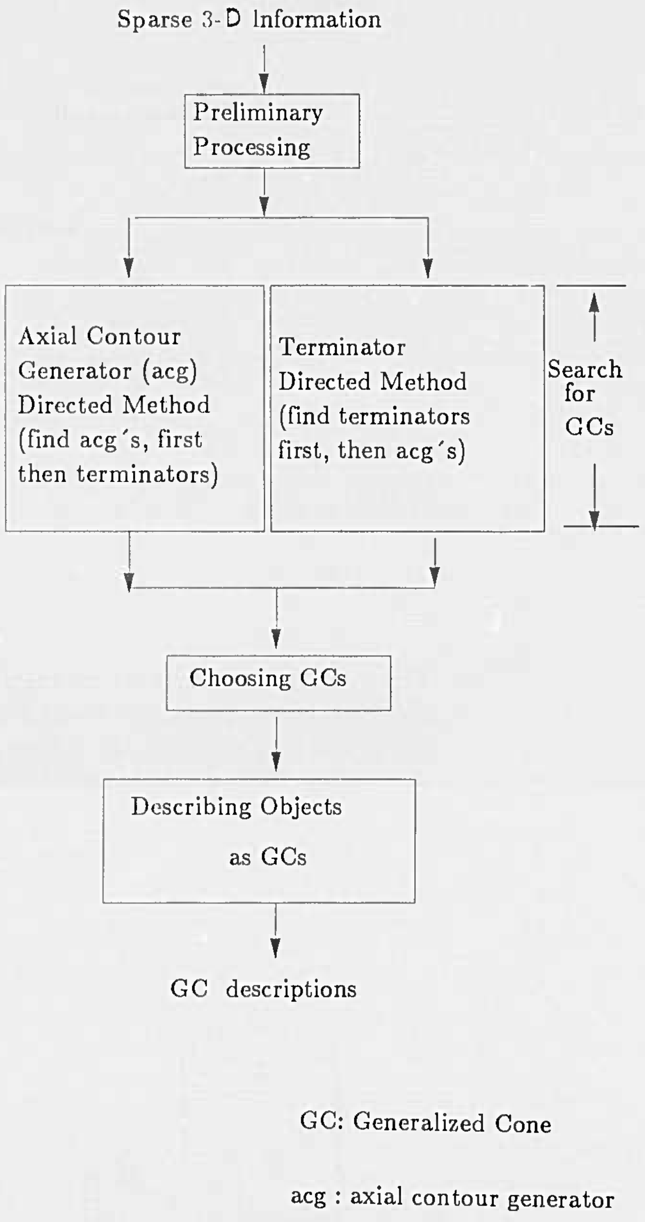

Sparse 3-D Information
Preliminary Processing
Axial Contour Generator (acg) Directed Method (find acg's, first then terminators)
Terminator Directed Method (find terminators first, then acg's)
Search for GCs
Choosing GCs
Describing Objects as GCs
GC descriptions
GC: Generalized Cone
acg : axial contour generator

Figure 3: Block diagram of the system

The system has been completely redeveloped from SAIL [11] on the DEC-20 system to LISP on a Symbolics 3600 series machine. In the redevelopment, we have modified our algorithms; we shall explain the modifications below.

While working with data available from a feature-based stereo system [12], we encountered noise and errors, that manifested themselves in two important properties of the line segments in the scenes: *planarity* and *smoothness of boundaries.* We found, by a simple technique, that segments that were expected to be coplanar were really not so. This is a serious problem because we require planarity for finding candidate axial contour generators and terminators. Therefore, we developed a least-squares method to evaluate planarity of a set of segments and to find the coefficients of the least-squares plane. The second point is that, contrary to our intuition that *a smooth curve in two stereo views should reconstruct to a smooth curve in 3-D*, we found that we actually obtained jagged contours as shown in figure 5. We believe that the jaggedness is primarily contributed by inaccuracies in edge detection and linear segment approximation. Error in these processes will cause about twice as much error in disparity computation, which in turn gets further exaggerated in depth calculation. More error will creep in because of inaccurate epipolar geometry and camera geometry. The jagged contours affect our system in tracing a contour and also in obtaining a smooth cross-section from the terminator. To remedy this problem, we trace the contour in 2-D in the image plane. To output a smooth cross-section we currently produce a circle, whose diameter is obtained from the distance between the axial contour generators.

Even when we used the least-squares technique, we found the errors in fitting planes to be very large. For example, for the cylinder in figure 5, we obtained an error of about 20 pixels for the terminator planes using the L_∞ norm. (The cylinder in the figure is about 80 pixels long.) We thus have problems in finding planar faces from stereo data and we cannot use the terminator directed method. We have, therefore, not yet redeveloped this in our system.

An important component of the system is **tracing a contour.** How do we trace a contour when we have surface marks, noise and missing data? As we discussed in our previous paper, standard transformation techniques like the Hough Transform cannot be used as we do not know the shape of the contour *a priori*.

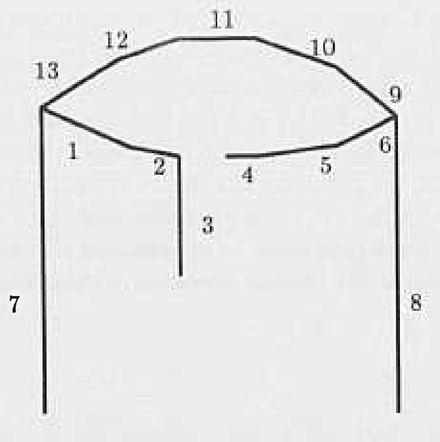


Figure 4: Problems with the earlier tracing algorithm

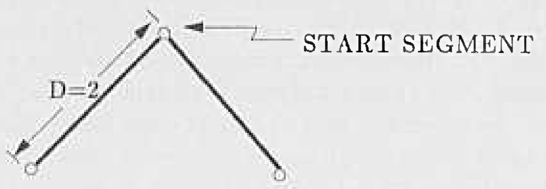
We have, therefore, developed a new algorithm. A previous version of the algorithm and associated background is given in [1]. Briefly, we treat the problem of tracing as the problem of finding a path from a start segment to a goal segment. In such a treatment, finding the next segment from the current path is of crucial importance. Previously, to find the next segment, we preferred connectivity over continuity and we handled the case of several segments connected to the current one before handling the case of missing segments. This is not always right as illustrated by the example in figure 4. The older version of the tracing algorithm (tracer) would pick up segment 3 after segment 2, which is erroneous. The other weakness of the previous tracer was that it would make hard and fast decisions at each point, coming up with a contour if successful. This would give incorrect results, if a wrong decision was taken at any point in the tracing.

To rectify these problems we have changed our tracing algorithm. We now handle the cases of missing segments and connected segments in a uniform way. The new algorithm has been implemented as an exhaustive depth first search (with a bound on the maximum depth searched and a limit on the branching factor of the search tree). Thus all possible contours from the start to the goal segment are found. The contours are then evaluated for smoothness. The smoothness criterion we use is based on local continuity. The smoothest curve is returned as the contour.

In our depth first search tree we need to decide on how to expand each node in the tree. This corresponds to deciding on what is (are) the next segment(s) from the current segment. The number of possible different next segments depends on the branching factor (B) we allow for our depth-first search tree (say 2). To choose the most appropriate next segment(s), we search in a neighbourhood of the current segment for segments satisfying some constraints and preferences. These are based on continuity and proximity and were described in detail in [1]. The tracing algorithm has also been modified to trace in 2-D in the image plane.

The new tracing algorithm has been found to give much better results for contours with a lot of missing segments and a lot of noise. It is exhaustive and tries all possibilities; it does not make a commitment at any point like the way the previous algorithm did. Also, this algorithm is very versatile, as different results can be obtained by changing the preference criteria for the next segment, the branching factor of the search tree and the way smoothness is evaluated. The algorithm has the problem that it could be very time consuming if we had long contours with several branches. If D is our depth bound and B the branching factor then the number of possible contours searched is equal to the number of leaves in the depth-first-search tree, $(B)^{D-1}$. If

$D = 10$ and $B = 2$ we have $2^9 = 512$ possible contours in the worst case. For example in the case below,



the number of contours $= 2^{D-1} = 2^{2-1} = 2$.

In the next section we shall show results with stereo data and data from an active range finder system of scenes with Linear Straight Homogeneous Generalized Cones (LSHGCs).

4 Results on real data for LSHGCs

The system was tested on scenes with synthetic data (as reported in [1]) and real data. We shall discuss and demonstrate results for the axial contour generator directed method but not for the terminator directed method, which has not yet been redeveloped.

We now present results with real data available from a segment-based stereo system [12]. Figure 6 shows the case of a single cylinder and figure 7 shows two objects with occlusion. In both cases we display the stereo pair of images, the segments extracted and the stereo output in two projections. Note, that the stereo data is very jagged. The output of our program is shown, with the cross-sections output as circles as discussed earlier.

For the cylinder case (figure 6) we have 45 segments and there are potentially $\binom{45}{2} = 990$ possible acg-pairs. Of these, the program explores 36 cases (by pruning the search space) and finds one object with acgs 17 and 33 and corresponding terminators. Other combinations are also found, like one with acgs 44 and 33, but they have lower ratings and are also part of the first combination. They are, therefore, eliminated and are not considered objects. The program takes about five minutes to run on this scene.

For the cylinder and frustum example (figure 7), we have 60 segments and there are potentially $\binom{60}{2} = 1770$ possible acg-pairs. Of these the program explores 105 cases and finds two objects: a combination with acgs 54 & 41 and corresponding terminators, and another with acgs 35 & 24 and corresponding terminators. Other combinations are also found, like one with acgs 41 & 24 but are eliminated because they have segments belonging to combinations with higher ratings. The program takes about 15 minutes to run on this scene. Note that no special and extra techniques have yet been developed to handle occlusion other than the ones mentioned in [1].

To demonstrate the generality of the system, we shall show results on imperfect boundaries extracted from active range-finder data. Curvature boundaries are extracted using the system described in [13] and then segments are fit and depth values assigned using the depth map. These 3-D line segments (and their corresponding edgels) are fed to our system.

An example of a simple cylinder is shown in figure 8. There are 19 segments in the figure, and thus there are potentially $\binom{19}{2} = 171$ acg-pairs, of which the program explores only 10 cases. The best combination, with 6 & 12 as acgs and corresponding terminators, is declared as the object found. Other combinations, like one with acgs 15 & 16, are also found but are eliminated as they have a lower rating and belong to the object already found. The program takes about one minute to run on this scene.

The above examples clearly demonstrate that our system is capable of handling textureless scenes with little information except at the boundaries, which too may have missing segments. That is, we are capable of working with very sparse (and imperfect) 3-D data, unlike previous researchers [2,3,7,6,8].

5 Conclusion and future work

We have developed a system for shape description from sparse, imperfect 3-D data with surface markings and missing data. We have demonstrated the working of the system on real data with Linear Straight Homogeneous Generalized Cones (LSHGCs) and some occlusion.

As part of future work, we intend to study complex generalized cones. By a complex generalized cone we mean a GC whose axis is not straight or whose sweep rule is not linear. The cross-section is still kept homogeneous, i.e., its shape does not change. Thus the complex GC is not an LSHGC, but could be an NLSHGC (Non-Linear, Straight Homogeneous GC) or an LNSHGC (Linear, Non-Straight Homogeneous GC) or an NLNSHGC (Non-Linear, Non-Straight Homogeneous GC) etc. In addition to the general properties of "in-betweenness" and tangency, which hold for the terminators and the ends of the axial contour generators (acgs), we have the important property of *smoothness* of the acgs. That is, each acg is a smooth contour. Apart from this, we conjecture that the axial contour generators of the complex GCs are piecewise coplanar. We intend to use the above properties in our analysis of complex generalized cones in the further development and extension of our system.

References

[1] Kashipati Rao and R. Nevatia. Generalized cone descriptions from sparse 3-d data. In *Proceedings of the DARPA Image Understanding Workshop*, pages 497–505, December 1985.

[2] W.E.L. Grimson. *From Images to Surfaces: A Computational Study of the Human Early Visual System.* MIT Press, 1981.

[3] D. Terzopoulos. *Multiresolution Computation of Visible-Surface Representations.* PhD thesis, Massachusetts Institute of Technology, Departments of Computer Science and Electrical Engineering, January 1984.

[4] T. O. Binford. Visual perception by computer. In *IEEE Conference on Systems and Controls*, Miami, December 1971.

[5] R. Nevatia. *Machine Perception.* Prentice Hall, 1982.

[6] G. J. Agin. *Representation and Description of Curved Objects.* PhD thesis, Stanford University, October 1972.

[7] R. Nevatia and T.O. Binford. Description and recognition of complex-curved objects. *Artificial Intelligence*, 8:77–98, 1977.

[8] D. Marr. Analysis of occluding contour. In *Proceedings of the Royal Society of London*, pages 441–475, 1977.

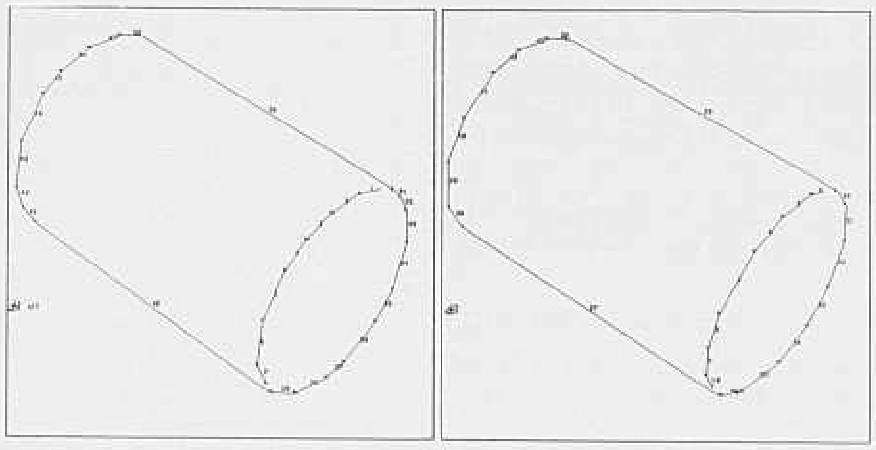
[9] R.A. Brooks. *Symbolic Reasoning among 3-D Models and 2-D images.* Technical Report AIM-343, Stanford Artificial Intelligence Laboratory, June 1981.

[10] S. A. Shafer. *Shadow Geometry and Occluding Contours of Generalized Cylinders.* Technical Report CS-83-131, Carnegie-Mellon University, May 1983.

[11] J. F. Reiser. *SAIL.* Technical Report STAN-CS-76-574, Department of Computer Science, Stanford University, August 1976.

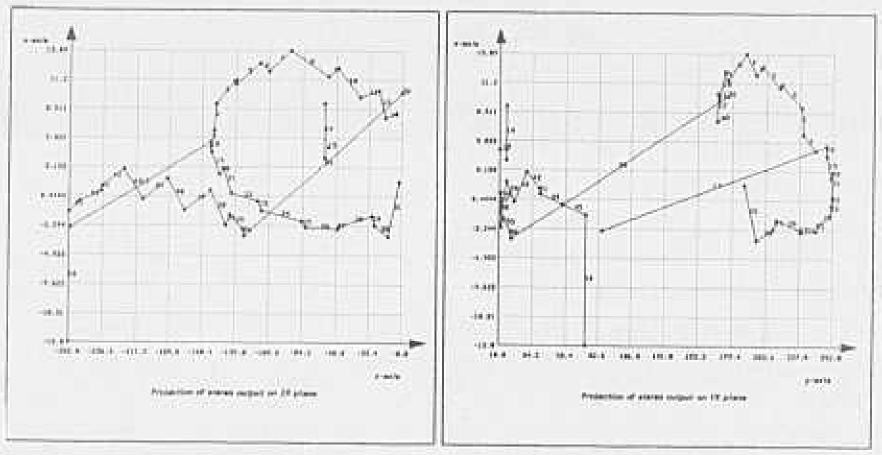
[12] Gerard Medioni and Ramakant Nevatia. Segment-based stereo matching. *Computer Vision, Graphics, and Image Processing*, 31:2–18, 1985.

[13] T. J. Fan, G. Medioni, and R. Nevatia. Description of surfaces from range data using curvature properties. In *Proceedings of Computer Vision and Pattern Recognition Conference*, pages 86–91, June 1986.



(a) (i) right (a) (ii) left

Smooth contours in a stereo pair of views

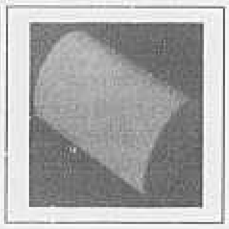


(b) (i) projection on zx plane (b) (ii) projection on yx plane

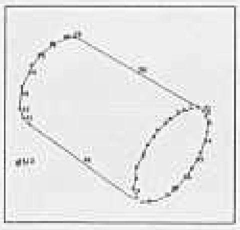
Jagged contours obtained after stereo (x-axis is depth)

Figure 5: Figure showing the jagged 3-D contours obtained from smooth 2-D contours in a stereo pair of views

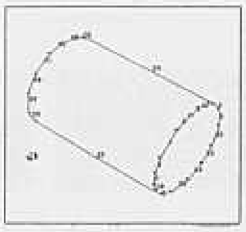
(a) (i) right (r)



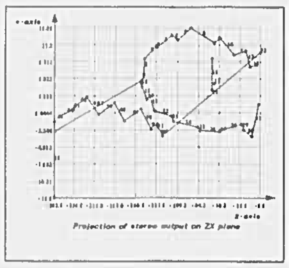
(a) (ii) left (l)



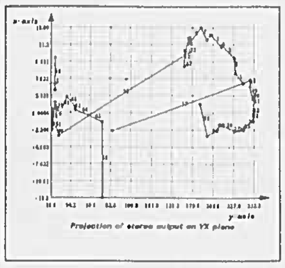
(b) (i) segments—r



(b) (ii) segments—l

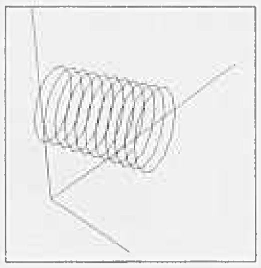


(c) (i) zx plane



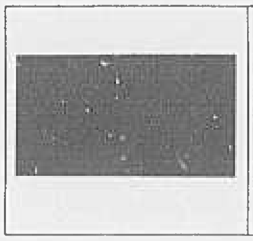
(c) (ii) yx plane

Projections of stereo output (x-axis is depth)

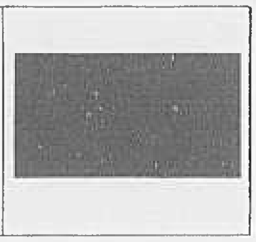


(d) output of our system

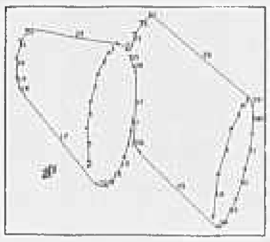
Figure 6: Data of a single cylinder from a passive range finding system (stereopsis) and output of our system



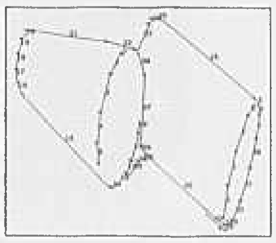
(a) (i) right (r)



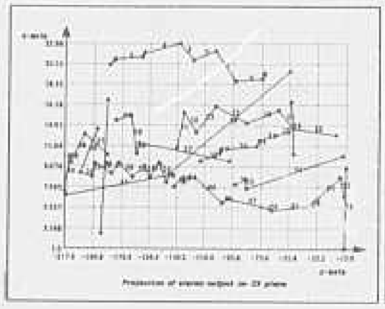
(a) (ii) left (l)



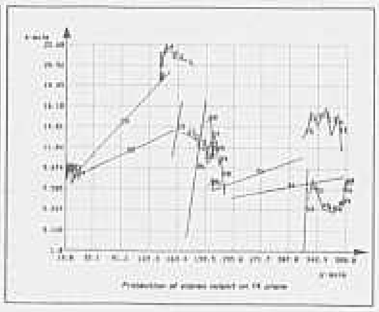
(b) (i) segments—r



(b) (ii) segments—l

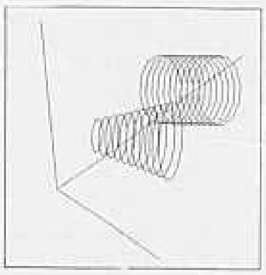


(c) (i) zx plane



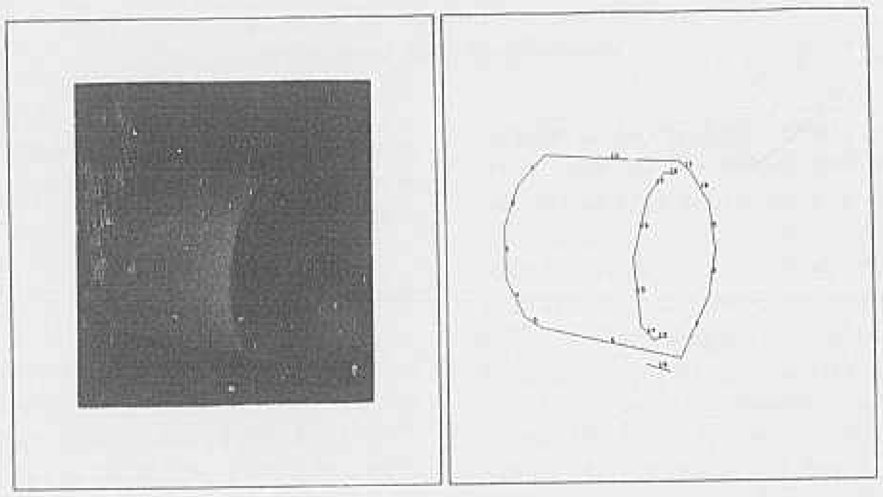
(c) (ii) yx plane

Projections of stereo output (depth along x-axis)

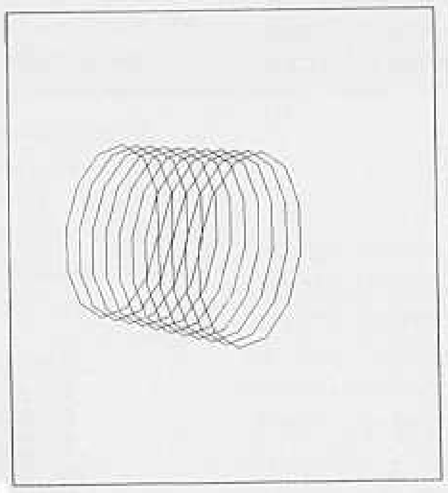


(d) output of our system

Figure 7: Data of two objects with occlusion from stereopsis and output of our system



(a) range image (b) segments from range image



(c) output of our system

Figure 8: Data of a simple cylinder from an active range finder (after processing using the system in Fan et al '86 and segment fitting) and output of our system.

Object Recognition Using Alignment

Daniel P. Huttenlocher
Shimon Ullman
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Abstract. This paper presents an approach to recognition where an object is first *aligned* with an image using a small number of pairs of model and image features, and then the aligned model is compared directly against the image. For instance, the position, orientation, and scale of an object in three-space can be determined from three pairs of corresponding model and image features. By using a small fixed number of features to determine position and orientation, the alignment process avoids structuring the recognition problem as an exponential search. To demonstrate the method, we present some examples of recognizing flat rigid objects with arbitrary three-dimensional position, orientation, and scale, from a single two-dimensional image. The recognition system chooses features for alignment using a scale-space segmentation of edge contours. Finally, the method is extended to the domain of rigid objects in general.

Introduction

Object recognition involves identifying a correspondence between part of an image and a particular view of a known object. This requires matching the image against stored object models to determine if any of the models could produce a portion of the image. Even for a single model, a given object can appear very different depending on its position and angle with respect to the viewer. First, from a particular view part of an object will generally be occluded. Second, an object may be distorted by projection into the image plane (e.g., forshortening). Finally, an object may itself undergo transformations such as having parts which move independently, or being stretched or bent. Most recognition systems assume that objects are rigid, and do not undergo any transformation [12] [4] [9] [2]. Some systems allow for perspective projection [16], and some have parameterized models which can articulate at certain points [7].

The presence of more than one object in an image also complicates the recognition problem. First, objects may occlude one another. Second, different objects in the image must somehow be individuated. In the case of touching and overlapping objects this generally cannot be done prior to recognition, but rather must be part of the recognition process itself.

The Task

In this paper we consider the problem of matching a two-dimensional view of an object against a potential model. The viewed object can have arbitrary three-dimensional position, orientation, and scale, and may be touching or occluded by other objects (3D from 2D recognition). First we consider the domain of flat rigid objects such as the widget shown in Figure 1. While the viewed object is flat, the problem is not two-dimensional because a flat object positioned in three-space can undergo distortion such as forshortening when projected into the image plane.

Like the general recognition task, this task suffers from problems of occlusion and of individuating multiple objects in an image. There is also a limited kind of shape distortion caused by projecting a rigid object into the image. We then consider extending the recognizer to the domain of rigid objects in general, such as the personnel carrier show in Figure 2.

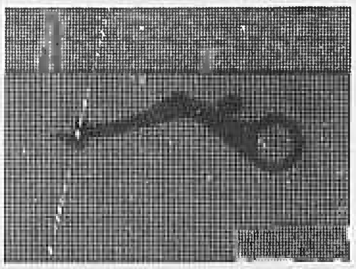


Figure 1. A flat widget used in recognition.

The current task cannot be handled by recognition systems which assume rigid objects with no distortion [12] [4] [9] [2], or by systems which only allow parameterized variation of rigid models [7]. The task is similar to that of Lowe, who addresses the problem of three-dimensional recognition from a single two-dimensional view [16]. However the task considered by Lowe is more restricted, because it is

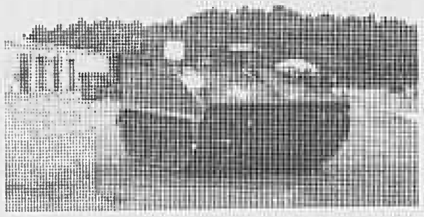


Figure 2. A personnel carrier used in recognition.

assumes that objects are polyhedral, and are viewed such that parallel surfaces appear more or less parallel.

In order to solve this task, we present a new approach where the recognition process is divided into two stages. In the first stage, a rigid object is *aligned* with an image using a small number of model and image features. In the second stage, the alignment is used to transform the model into image coordinates. The key observation underlying the alignment operation is that the position and orientation of a rigid object can be determined from a small number of position and orientation measures. Once the position and orientation have been determined, the model can be compared directly with the image. In contrast, current recognition systems search for the largest set of model and image feature pairs which are consistent with a single position and orientation of a rigid object. The number of such sets is exponential, requiring the use of various techniques to limit the search.

Matching Models and Images: Previous Approaches

In this section we briefly discuss the limitations of some recent recognition systems with respect to the recognition task described above (for a more general review see [3]). These systems all exploit rigidity by noting that for a given position and orientation of a rigid object, there must be a single transformation which maps each model feature onto its corresponding image feature. This transformation consists of a three-dimensional rotation and translation in 3D from 3D recognition, and a solution to the perspective viewing equation in 3D from 2D recognition [16].

Recognition is generally structured as a search for the largest pairing of model and image features for which there exists a single transformation mapping each model feature to its corresponding image feature [12] [4] [16] [7] [9]. For i image features and m model features there are at most $p = i \times m$ pairs of model and image features. Because of occluded image points, and image points which do not correspond to the model, any subset of these p pairs could be the largest set of matching model and image points, and thus the number of possible matches is exponential in the size of p. Two methods are used to limit this space of possible matchings of model and image features.

The first method of limiting the possible matches is to use the identity of features to restrict the pairing of model and image features. However, even in the ideal case where each model feature has only a single corresponding image feature, there may be multiple matches to consider because of image features which actually correspond to other objects in the scene. Shape descriptions such as SLS [6], codons [14], and the curvature primal sketch [1] are all intended to produce relatively unique features for use in recognition.

The problem with using the identity of features in recognition is that there is a tradeoff between the uniqueness of a feature and the robustness with which it can be recognized. Since systems which rely heavily on the identity of features must use relatively unique features, they tend to be sensitive to noise and occlusion in the image.

For instance, the LFF [4] and 3DPO [5] recognition systems form feature descriptions by using local clusters of features. A "focus feature" in each cluster is chosen for use in matching. This feature is described in terms of its type (e.g., corner, hole), and the type, distance, and angle of the other features in the cluster. The use of local feature clusters yields relatively unique features. However, it is difficult to ensure that each cluster is composed of features from a single object, making the system sensitive to the position and orientation of neighboring and occluding objects.

The second method of limiting the possible matches is to use relations between features to eliminate inconsistent pairs of model and image features [12] [9] [4]. For instance, in order for two pairs of model and image features (m_1, i_1) and (m_2, i_2) to be part of a consistent set, the distance between the image features i_1 and i_2 must be the same as the distance between the model features m_1 and m_2, within some error bound. Similarly, the angle between orientation measures for any pair of image features must match the angle between the corresponding pair of model features.

The problem with using relations between features in recognition is that the relations must be measurable in the image. Since relations such as distance and angle are not invariant under projection, three-dimensional recognition systems which use these relations require three-dimensional data. Relations which are invariant under projection tend to be much weaker than distance and angle relations.

It has been demonstrated that distance and angle relations can be used to greatly limit the number of possible matches of a model to an image, for both 2D from 2D [12] [4] and 3D from 3D [13] [5] recognition tasks. However, the method cannot readily be extended to handle 3D from 2D tasks. In addition to empirical demonstrations of the power of distance and angle relations among features, it has been shown that simple bounded-noise distance constraints can be used to develop an $O(n^2)$ time algorithm for recognizing an isolated object with n features which can undergo two-dimensional rotation and translation. [2].

The SCERPO system [16] [17] is the only one to address the problem of three-dimensional recognition from a single two-dimensional view. Image edges are grouped together using proximity and parallelism. These local groupings are then used to form pairs of model and image features. A given set of feature pairs are consistent if there is a solution to the perspective viewing equation which maps each model feature onto its corresponding image feature. There are no simple pairwise checks such as distance and angle relations which can be used to incrementally test the consistency of a set of pairs given an added pair. Instead, the perspective viewing equation is solved for each added pair of model and image points. This is done using Newton-Raphson iteration, which allows a solution to be modified to account for a new pair.

None of these recognition systems effectively address the task of recognizing objects which have arbitrary three-dimensional position, orientation and scale, from a single two-dimensional view. Each system restricts the recognition task in order to limit space of possible matches of models and images (e.g., using distance and angle relations, or requiring parallel surfaces to appear parallel). In the next section we present a new approach for matching models and images which does not structure recognition as an exponential search. We then demonstrate this method on several images.

The Alignment Method of Recognition

We have seen above that recognition can be viewed as a search through the space of all possible positions and orientations of all possible objects. The idea of the *alignment* approach is to separate this search into two stages. In the first stage, the position, orientation, and scale of an object are found using a minimal amount of information, such as three pairs of model and image points. In the second stage, the alignment is used to map the object model into image coordinates for comparison with the image.

There are two major advantages of this approach. First, by using a small fixed number of model and image features, we avoid structuring recognition as a search through an exponential space. Second, a given alignment can be used to match multiple models against the image. If a group of objects are stored such that they are aligned with one another, then a single alignment computation will map the entire group of objects into the image.

The key observation behind the approach is that the alignment can be performed with a small amount of information. For example, three image points and three corresponding model points are sufficient to determine the position, orientation and scale of a rigid object in three-space. Similarly, two points and an orientation measure can also be used to solve for the three-dimensional alignment.

In the case of two-dimensional recognition, only two pairs of corresponding model and image points are needed to align a model with an image. Consider two pairs, (m_1, i_1) and (m_2, i_2), such that model point m_1 corresponds to image point i_1 and model point m_2 corresponds to image point i_2. First the model is translated in x and y such that m_1 is coincident with i_1. Then it is rotated about the new m_1 such that the edge m_1m_2 is coincident with the edge i_1i_2. Finally the scale factor is computed to make m_2 coincident with i_2. These two translations, one rotation, and a scale factor will make each unoccluded point of the model coincident with its corresponding image point, as long as the initial correspondence of (m_1, i_1) and (m_2, i_2) was correct.

For 3D from 2D recognition, the alignment method is somewhat more complicated. In the final section of the paper, we show how to use three pairs of model and image points to position and orient a model in three-space given a single two-dimensional view, assuming orthographic projection. A transformation from the model to the image consists of 2-dimensional translation, three-dimensional rotation, and a linear scale factor which is proportional to the viewing distance. Under normal viewing conditions, orthographic projection plus scale is a reasonable approximation to perspective viewing. Under high perspective distortion – when the object occupies much of the field of view – the approximation is poor. However, under such conditions people also appear to be bad at recognition.

Consider an object, O, with three-dimensional positional freedom, and a two-dimensional image, I, which contains a view of O (perhaps along with other objects). We are interested in using the alignment computation to find O in the image. The feature detector discussed in the next section returns a set of pairs of matching object and image features, P. Any three-tuple of the pairs in P specifies a possible alignment of the object with the image. In general some small number of these alignments will define the true position and orientation of the object, and the rest will be due to incorrect matchings of model and image points. Thus the recognition problem is to determine which alignment in P defines the transformation mapping the most model points onto their corresponding image points.

Given a set of pairs of model and image features, P, we solve for the alignment specified by each triple in P. For some triples there will be no possible alignment of the three model and image features. Each remaining triple specifies a transformation mapping model points to image points. An alignment is scored by using the transformation to map the model edges into the image, and correlating the transformed model edges with the image edges. The best alignment is the one which correlates the most model edges with image edges.

For m model features and i image features, the number of pairs of model and image features, p, is at most $i \times m$. With a good labeling scheme, the number of pairs, p, will be much smaller, approaching m when each model point has one corresponding image point. Given p pairs of features, there are $\binom{p}{3}$, or $O(p^3)$, three-tuples of pairs, each of which specifies a possible alignment of the model and the image. Each alignment is scored by mapping the model edges into the image. If the model edges are of length l, then the worst case running time of the algorithm is $O(lp^3)$. Thus by structuring the recognition process as an alignment stage followed by a comparison stage, it is transformed from the exponential problem of finding the largest consistent set of model and image points, to the polynomial problem of finding the best triple of model and image points.

Shape Features: Segmenting Edge Contours

The recognition system uses edge-based shape features for matching models against images. The input to the system is a grey-level image. The image is processed by an edge detector [8], and the edges are chained together using an eight-way chaining algorithm. Chains with low overall edge strength are discarded. Thresholding whole chains rather than individual edge points produces a more stable output. Finally, edge chains with unambiguous nearest neighbors are merged together if they can be connected by a smooth spline.

Once pieces of edge contour have been chained together, simple shape descriptors are derived using the local curvature of the edge contours. The curvature is computed as the change in angle (per unit arclength) between local tangent vectors at neighboring pixels. The tangents are computed using the least squares best fit line over a small local neighborhood.

A relatively stable segmentation of an edge contour is obtained by breaking the contour at zero crossings of curvature (inflection points in the contour). Zero curvature regions are also identified, segmenting the contour into straight, positive curvature and negative curvature pieces. Zero crossings of curvature were chosen as segmentation points because they are stable under three-dimensional rotation and projection, except in the degenerate case where they disappear (when the contour projects into a line). Maximum curvature points, which are often used to segment edge contours [10] [16], are not at all stable under rotation and projection.

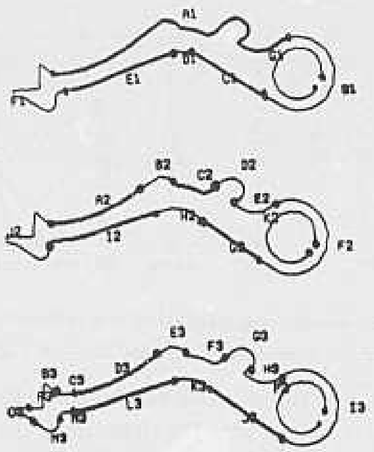


Figure 3. A simple curvature scale-space segmentation of a widget.

A hierarchy of curve segmentations can be obtained by smoothing the curvature at different scales, and using the smoothed curvature to segment the edge contour. Smoothing in curvature space preserves only those inflection points in the edge contour (zero crossings of curvature) which are significant at a given scale. Thus coarser scale segments correspond to merging neighboring segments at finer scales, producing a scale-space [18] [19] hierarchy of segments. Since coarser scales of smoothing do not introduce zero crossings which were not present at finer scales, the hierarchy forms a tree of segments from coarser to finer scales.

Figure 3 shows a three-level scale-space curvature segmentation of the edge contours of the widget in Figure 1. Each part of the Figure shows the same contour, segmented according to the curvature smoothed at different scales (using Gaussian filters of size $\sigma = 7$, 20, and 40 pixels, respectively). The coarsest scale is at the top of the Figure and the finest scale is at the bottom. The endpoints of each segment are delimited by a dot, and zero curvature regions are shown in bold.

Each segment of edge contour is classified according to whether it is curved or `straight`. The curved segments are further classified by the degree of closure: open or `closed`, and the smoothness of the contour: `smooth` or `unsmooth`, yielding a total of five types of segments. Richer descriptions are then obtained by combining the classifications at multiple scales of smoothing.

The three segmentations in Figure 3 form a (multi-rooted) tree, where each region at a coarse scale corresponds to one or more regions at each finer scale. Figure 4 shows this scale-space segment tree. Each segment in the tree is indicated by its label from Figure 3, and by the type of segment: `straight`, `curve`, and `open-curve`.

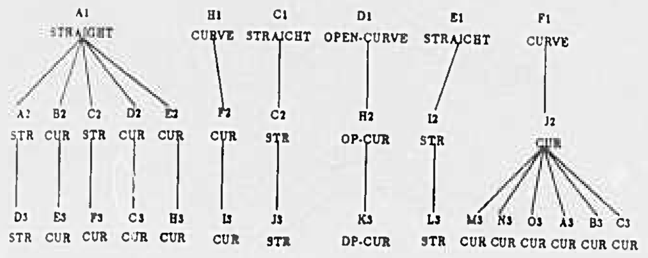


Figure 4. Multi-scale segmentation tree corresponding to the curvature scale-space segmentation in Figure 3.

Multi-scale descriptions are formed using the types of segments at a given scale, plus the structure of the hierarchy at the next finer scale. Two aspects of the tree structure are used. First, a segment is classified according to whether it corresponds (primarily) to one or many segments at the next finer scale: `single` or `multiple`. Second, a `multiple` segment is classified according to whether or not the finer scale segments form a regular pattern. A pattern consists of a repetition of the same type of segment, in either the same or opposite directions of curvature. This yields the classification `irregular`, `regular-same`, and `regular-opposite`.

For example, using this multi-scale description, the `straight` segment `A1` at level 1 in the tree is differentiated from the other `straight` segments `C1` and `E1` at the same level, because `A1` is composed of `multiple`, `irregular` segments at the next level whereas `C1` and `E1` are each composed of a `single` segment.

Using the multi-scale description, at the coarsest scale the widget is composed of seven segments, only two of which can be confused with one another (the two straight segments `C1` and `E1`). The segments are, `A1`: `straight`, `multi`, `irregular`; `B1`: `curve`, `single`, `smooth`; `C1`: `straight`, `single`; `D1`: `open-curve`, `single`; `E1`: `straight`, `single`; `F1`: `curve`, `multi`, `irregular`; `G1`: `closed-curve`, `single`, `smooth`. Thus, the result of this multi-level description is relatively unique classification of the edge segments, even though the individual features are coarse, and sparsely distributed in the image.

Points for use by the alignment algorithm are obtained from the features as follows. For closed segments of contour, the center of the region defined by the contour is used. This point is found from the intersection of the major and minor axes of the region. For straight segments the endpoints are used, and for other curved segments the middle of the curve is used. Since the endpoints of the segments are at inflection points or the ends of zero curvature regions, they are relatively stable, making it reasonable to use endpoints and midpoints for matching.

Given a model and an image, the multi-scale description of both the model and the image are computed. The coarse level descriptions are then used to determine which model features can potentially match which image features. These possibly matching features form the set of pairs, P, for the alignment algorithm. In scoring how well an alignment matches an image, fine scale model features are used.

Examples

In this section we show several examples of the recognizer processing grey-scale images of widgets. The model is the multi-scale description of the widget shown in Figure 3 and Figure 4. The model is just the result of processing the image of the isolated widget in Figure 1 in the same manner as any image. Thus for flat objects, models are formed directly from an image of an object.

The recognizer is implemented on a Symbolics 3650, and takes from 2-5 minutes for each of the examples shown in this section. A multi-scale description of the edge segments is formed and used to define alignment points in the image, as described in the previous section. Possible alignments are then computed using all three-tuples of model and image point pairs. For each alignment, the model is mapped into the image and the transformed model edges are correlated with the image edges. The alignments are ranked based on the percentage of the model edge contour for which there is a corresponding image edge contour.

Each example is illustrated by a Figure with four parts: i) the grey-level image, ii) the thresholded image edges extracted by the edge detector, iii) the model edges, iv) the edges of the aligned model superimposed on the image. Part iv) also indicates the points which were used in computing the alignment of the model with the image.

The example in Figure 5 shows two widgets in the plane. The top widget has been flipped over, and thus cannot be recognized using only two-dimensional transformations. The recognizer finds two distinct positions and orientations of the model which match 99% and 98% of the model edge contour to image edges. These two matches are shown in part iv) of the Figure.

Another position and orientation is found at the alignment stage, but is eliminated because the correlation with the image is poor, and the image edges are accounted for by a better alignment. This alignment is shown in Figure 6 superimposed with the image edges. This alignment is found because the two straight edges are indistinguishable, and the three points used for alignment were the two straight edges and the bend.

Figure 7 shows a widget which has been tilted approximately 30 degrees by resting one end of it on a block, forshortening the image. The recognizer finds a single best position and orientation, which is shown in part iv) of the Figure.

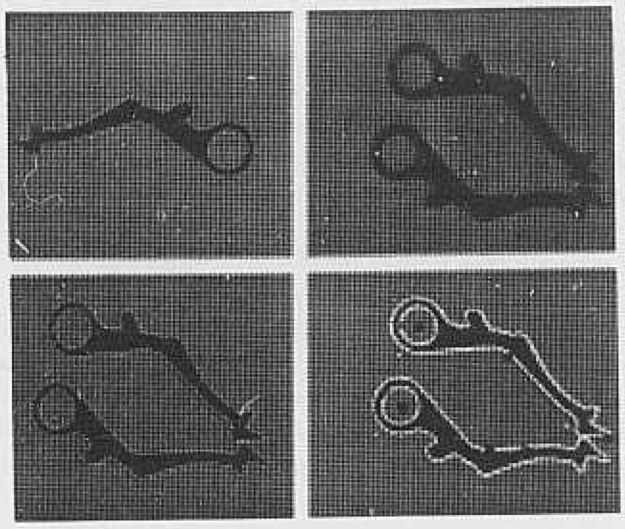


Figure 5. Matching a widget against an image of two widgets in the plane. See the text for a description.

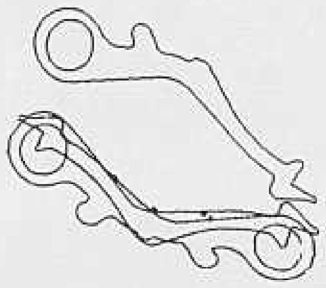


Figure 6. An alignment of a widget with an image which does not match the fine scale model features with image features.

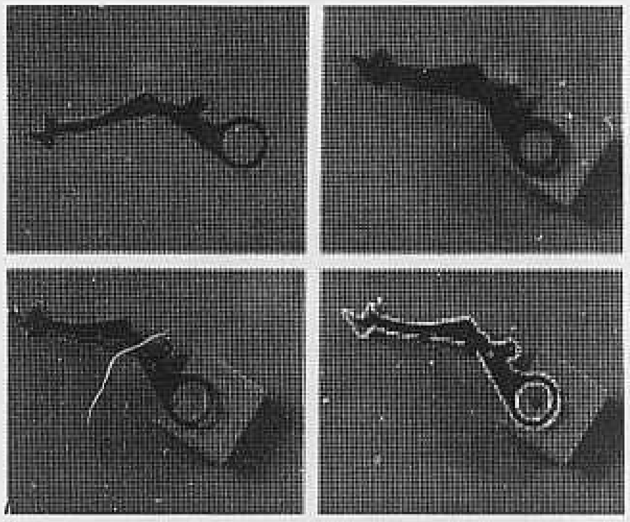


Figure 7. Matching a widget against an image of a forshortened widget.

The next example, in Figure 8, shows another widget which has been tilted out of the image plane — for instance the circular end is now an oval. The best match is shown in part iv) of the Figure.

Finally, we demonstrate the ability of the recognizer to find partly occluded objects. As long as three features are visible in the image, the alignment algorithm will be able to

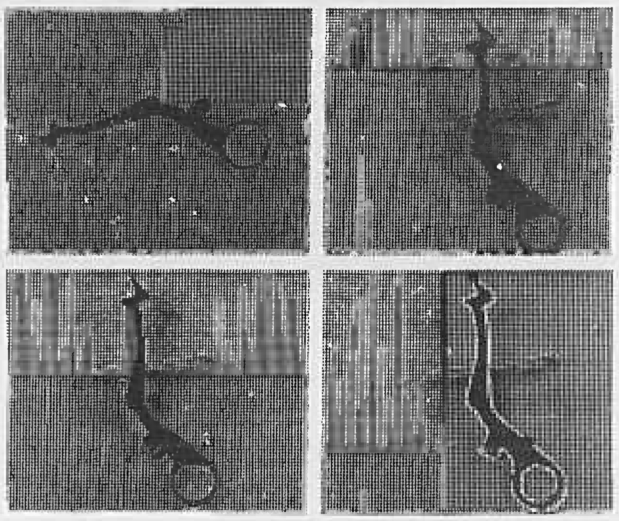


Figure 8. Matching a widget against an image of a tilted widget.

align the model with the image. Figure 9 shows a widget which has a pile of smaller widgets obscuring the circular end. The best alignment matches 80% of the model edge contour to image edges, and is shown in part iv) of the Figure. Figure 10 shows two widgets obscured by each other and several smaller objects. The matcher finds two distinct positions and orientations, which are shown in part iv) of the Figure.

From these examples we see that the alignment algorithm finds a small number of reasonable matches of widgets to images, even when the widget is forshortened, scaled, and partly occluded. The scoring method of mapping model edges into the image provides a simple method for finding the best alignment.

Recognizing Non-Flat Objects

Extending the alignment method to recognize non-flat objects such as the personnel carrier in Figure 2 is relatively straightforward. The only difference between using a three-dimensional model and a two-dimensional model is the need to eliminate portions of the object which are not visible from a given position and orientation. Thus, after computing a possible alignment, model edges and surfaces which are not visible from the specified position and orientation must be removed before the model is mapped into the im-

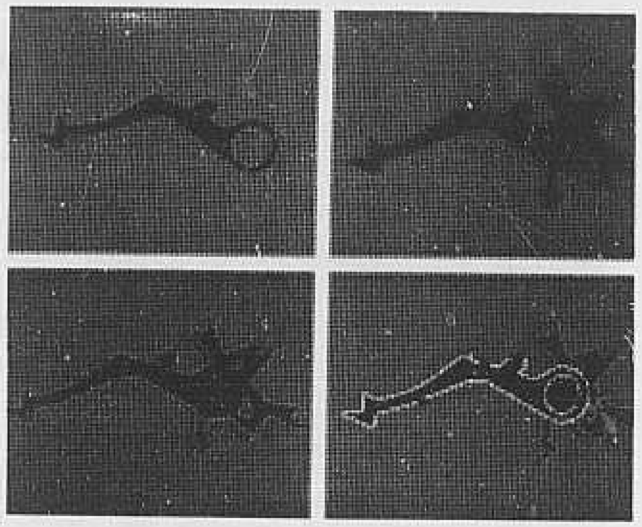


Figure 9. Matching a widget against an image of a partly occluded widget.

age (cf. [11]).

For a three-dimensional model, it must also be ensured that the three points used in alignment are all visible given the position and orientation of the object which they specify. Thus for each possible alignment, the three model points must be checked to make sure that they are not on hidden edges or surfaces.

Rather than using a single three-dimensional model of an object, it is possible to use multiple planar views, with one view for each position from which a different set of object surfaces are visible. Using multiple local alignments, such a planar model can be matched to a variety of different images of the object.

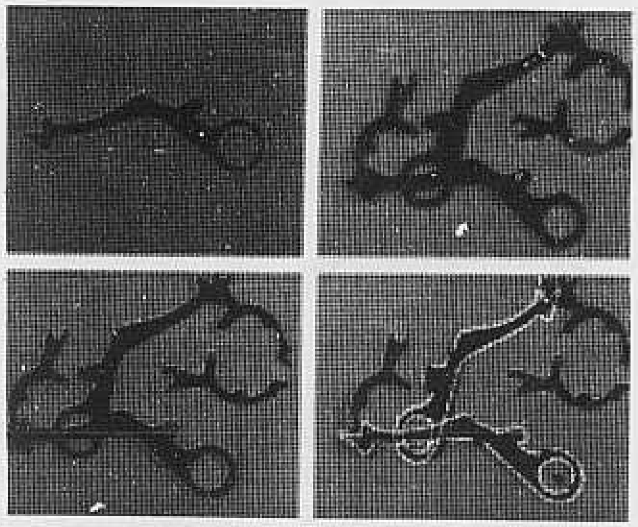


Figure 10. Matching a widget against an image of two partly occluded widgets.

Using just a single alignment, a planar model of a three-dimensional object will only match images which are taken at close to the same viewing angle as the model view. The distortion in using a planar model comes from the fact that the model actually contains multiple object planes, even though the points in the model are only two-dimensional. However by using multiple alignments, one for each plane of the object, a flat model can be matched to images from a wide range of different viewpoints.

One method of finding such regions for local alignment is to triangulate the set of model features used in alignment, and separately align each triangular region. To map the model to the image, all the model points inside a given triangle are transformed using the alignment for that triangle. To the extent that a triangle falls on a nearly planar part of an object, this will produce a correct alignment for that region. By aligning each locally neighboring triple of model features found in the triangulation, rigidity is preserved for

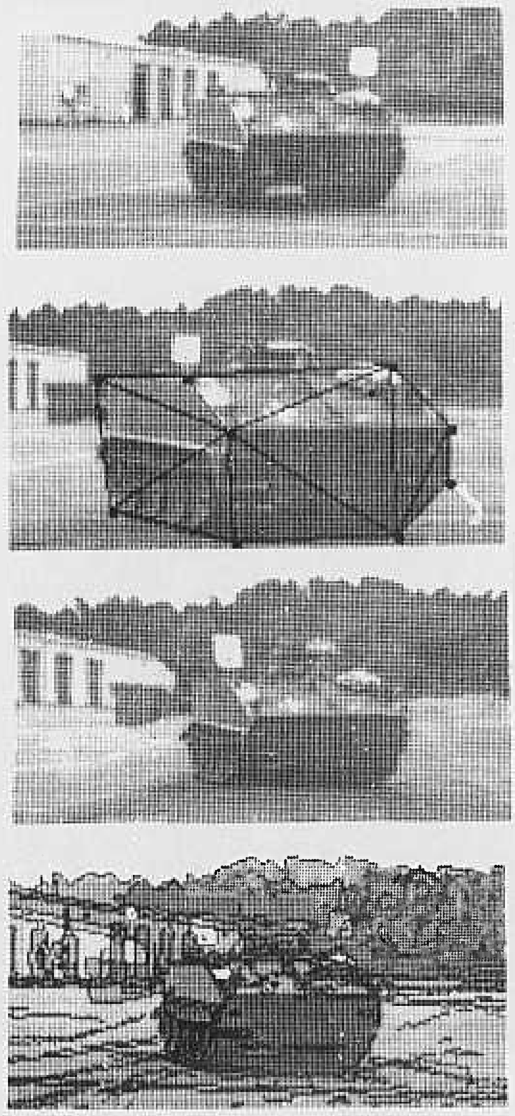


Figure 11. Local alignment of two views of a personnel carrier.

each triangle, but not for the object as a whole.

Points in a given triangle will not be mapped into the image correctly if they are not on the same plane of the object as the three points defining the triangle. However, such points will usually be transformed to a point near their correct position because the alignment will be partially correct. Therefore the locally-rigid alignment process can be iterated, starting with a two-point or three-point alignment, and using the partial match to pick potential corresponding model and image points for computing more refined, and less globally rigid, alignments.

To show how this local alignment algorithm works, the image of a personnel carrier in Figure 2 is matched with another view taken from a slightly different position, shown in Figure 11a. Simple three-dimensional alignment of these images is insufficient because the side of the personnel carrier has become substantially forshortened whereas the front has not. Figure 11b shows the triangulation used for matching the two images (the correspondence between the two images was specified by hand). Figure 11c shows a synthetic image formed by transforming the image in part b according to the specified triangles. Figure 11d shows the images in parts a and c superimposed. To better see how well the two personnel carriers match, the intensity edges of the two images are also superimposed on the images. The match is good enough to even bring small details such as the white cross painted on the front into correspondence.

The 3D from 2D Alignment Method

In this section we present the alignment method in detail. We show that the position, orientation, and scale of an object in three-space can be determined from a two-dimensional image using three pairs of corresponding model and image points. This method approximates perspective viewing by orthographic projection plus a scale factor which is proportional to the viewing distance. As mentioned above, this is a reasonable approximation except when the object is very large with respect to the viewing distance.

Consider three model points a_m, b_m and c_m and three corresponding image points a_i, b_i and c_i, where the model points specify three-dimensional positions, (x, y, z), and the image points specify positions in the image plane, $(x, y, 0)$. The alignment task is to find a transformation which maps the plane defined by the three model points onto the image plane, such that each model point coincides with its corresponding image point. If no such transformation exists, then the alignment process must determine this fact.

Since the viewing direction is along the z-axis, an alignment is a transformation which positions the model such that a_m projects along the z-axis onto a_i, and similarly for b_m onto b_i, and c_m onto c_i. The transformation consists of translations in x and y, and rotations about three orthogonal axes. There is no translation in z because all points along the viewing axis are equivalent under orthographic projection. Instead, distance from the viewer is reflected by a change in scale.

The first step in finding an alignment is to translate the model points so that one point projects along the z-axis onto its corresponding image point. Using the point a_m for this purpose, the model points are translated by $(x_{a_i} - x_{a_m}, y_{a_i} - y_{a_m}, 0)$, yielding the model points a'_m, b'_m and c'_m. This brings a'_{mi}, the projection of a'_m into the image plane, into correspondence with a_i, as illustrated in Figure 12a.

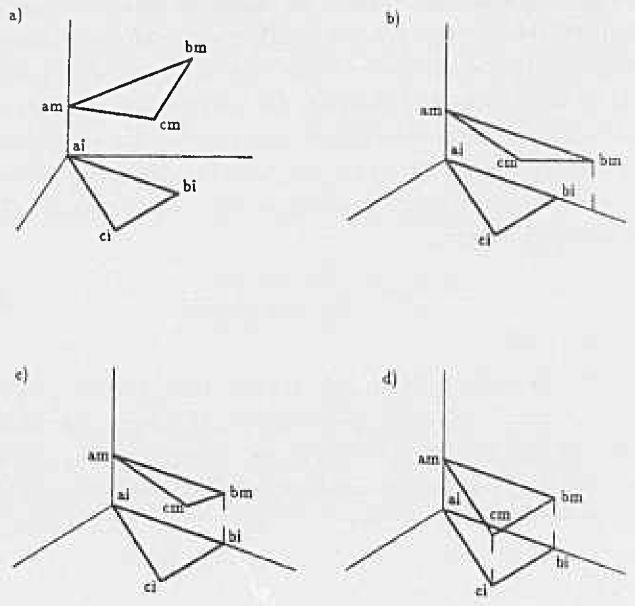


Figure 12. The alignment process: a) the points a_i and a_m are brought into correspondence, b) the ab edges are aligned, c) the points b_i and b_m are brought into correspondence, d) the points c_i and c_m are brought into correspondence.

Now it is necessary to rotate the model about three orthogonal axes to align b_m and c_m with their corresponding image points. First we align one of the model edges with its corresponding image edge by rotating the model about the z-axis. Using the $a'_m b'_m$ edge we rotate the model by the angle between the image edge $a_i b_i$, and the projected model edge $a'_{mi} b'_{mi}$, yielding the new model points b''_m and c''_m, as illustrated in Figure 12b.

To simplify the presentation, the coordinate axes are now shifted so that a_i is the origin, and the x-axis runs along the $a_i b_i$ edge.

Since b''_{mi}, the projection of b''_m into the image plane, lies along the x-axis, it can be brought into correspondence with b_i by simply rotating the model about the y-axis. The amount of rotation is determined by the relative lengths of $a_m b_m$ and $a_i b_i$, because the model must be rotated such that the projected model edge is the same length as the image edge. If the model edge is shorter than the image edge, then there is no such rotation, and hence the model cannot be aligned with the image.

Thus, the model points b''_m and c''_m are rotated about the y-axis by ϕ to obtain b'''_m and c'''_m, where

$$\cos\phi = \frac{\|b_i \cdot (1,0,0)\|}{\|b_m \cdot (1,0,0)\|}$$

for $0 \leq \cos\phi \leq 1$. Note that since a_i is the origin, the length of the edges is just b_i and b_m. The result of this rotation is illustrated in Figure 12c.

Finally, c'''_m is brought into correspondence with c_i by rotation about the x-axis. The degree of rotation is again determined by the relative lengths of model and image edges. However, in the previous case, the edges were parallel to the x-axis, and therefore the length was the same as the x component of the length. In this case, the edges need not be parallel to the y axis, and therefore the y component of the lengths must be used. Thus, the rotation about the x-axis is θ, where

$$\cos\theta = \frac{\|c_i \cdot (0,1,0)\|}{\|c_m \cdot (0,1,0)\|} \tag{2}$$

for $0 \leq \cos\theta \leq 1$.

If the model distance is shorter than the image distance, there is no transformation which aligns the model and the image. Furthermore, if the rotation does not actually bring c''''_{mi} into correspondence with c_i, then there is also no alignment. This latter case can result because the rotations are those which will bring the points into alignment if there actually is a consistent solution. If there is no solution then it may still be possible to solve for the rotations, but they will not bring all three points into alignment.

The final rotation brings the plane defined by the three model points into correspondence with the image plane, as illustrated in Figure 12d. This combination of translations and rotations can now be used to map the model into the image, in order to determine if the object is in fact in the image at this position and orientation.

Now we show how to solve for a linear scale factor as a sixth unknown. First we note that scaling the model changes only the final two rotations (about the x- and y-axis), and not the x and y translations, or the rotation around the z-axis. This is because scaling changes the distances between the three model points, and only the rotations about the x- and y-axis depend on the distances between points.

The final two rotations align b_m with b_i, and c_m with c_i. The alignment of b_m involves movement of b_{mi} along the x-axis, whereas the alignment of c_m involves movement of c_{mi} in both the x and y directions.

Since the movement of b_{mi} is a sliding of along the x-axis, only the x-component, x_b, changes. The change is given by the rotation ϕ about the y-axis, as in (1). With a scale factor, s, this becomes

$$x'_b = s x_b(\cos\phi) \tag{3}$$

Similarly the movement of c_{mi} in the y direction is given by the rotation θ about the x-axis, as in (2). With a scale factor this becomes

$$y'_c = s y_c(\cos\theta) \tag{4}$$

The movement of c_{mi} in the x direction is given by the rotations about both the x- and the y-axis. From the matrix for a combined rotation about the x- and y- axis we obtain

$$x' = (x\cos\theta + y sin\phi \sin\theta)$$

Thus with the scale factor, the the x component of c_m is

$$x'_c = s(x_c\cos\theta + y_c\sin\phi\sin\theta) \tag{5}$$

Now we have three equations in the three unknowns, s, θ, and ϕ. One method to solve for s is to substitute for $\cos\theta$, $\sin\theta$, and $\sin\phi$ in (5). From (3) we know that,

$$\sin\phi = \frac{1}{s x_b}\sqrt{s^2 x_b^2 - x_b'^2} \tag{6}$$

And similarly from (4),

$$\sin\theta = \frac{1}{s y_c}\sqrt{s^2 y_c^2 - y_c'^2} \tag{7}$$

Substituting (6) and (7) into (5) and simplifying yields

$$s^2(x_b x'_c - x_c x'_b)^2 = (s^2 x_b^2 - x_b'^2)(s^2 y_c^2 - y_c'^2)$$

Expanding out the terms we obtain

$$s^4(x_b^2 y_c^2) - s^2(\tau_b^2 y_c'^2 + x_b'^2 y_c^2 + (x_b x'_c - x_c x'_b)^2) + x_b'^2 y_c'^2$$

a quadratic in s^2. While there are generally two possible solutions, it can be shown that only one of the solutions will specify possible values of $\cos\phi$ and $\cos\theta$.

Having solved for the scale of an object, the final two rotations ϕ and θ can be computed using (1) and (2) modified to account for the scale factor,

$$s(\cos\phi) = \frac{\|b_i \cdot (1,0,0)\|}{\|b_m \cdot (1,0,0)\|}$$

and

$$s(\cos\theta) = \frac{\|c_i \cdot (0,1,0)\|}{\|c_m \cdot (0,1,0)\|}$$

Thus solving for scale involves the computation of s, and slight a modification to the computation of the final two rotations.

For planar models, the three-dimensional alignment task only involves mapping points from one plane to another. Therefore, a planar model can be aligned with an image using only two-dimensional operations. In effect, the actual three-dimensional rotation and translation of the object are simulated in the plane. This *planar alignment* specifies x and y translation, planar rotation, x scale, y scale, and a shear which is a scaling in x proportional to distance from the y-axis (or vice verse). The details of this alignment method are given in [15].

Summary

Recognition is generally viewed as a search through the space of possible positions and orientations of objects. The idea of the *alignment* approach is to separate this search into two stages. In the first stage, the position, orientation, and scale of an object are found using a minimal amount of information, such as three pairs of model and image points. In the second stage, the alignment is used to map the object model into image coordinates for comparison with the image.

There are two major advantages of this approach. First, by using a small fixed number of model and image features, we avoid structuring recognition as search through an exponential space. Second, by storing object models such that they are aligned with one another, a single alignment computation can be used to map multiple objects into the image.

The key observation behind the approach is that the alignment can be performed with a small amount of information. For example, three points are sufficient to determine the position, orientation and scale of a rigid object in three-space from a single two-dimensional image. Similarly, two points and an orientation measure can also be used to solve for this alignment.

We have implemented a recognizer using the alignment method. This system chooses features for alignment using a scale-space segmentation of edge contours. Coarse scale segments are described both in terms of their shape, and the structure of the scale-space hierarchy at the next finer level. This produces relatively unique features for use in finding possible alignments of a model and an image. Fine scale segments are then used to determine how well a given alignment matches the model with the image.

To demonstrate the recognition method, several examples were shown of the recognizer finding a widget with arbitrary three-dimensional position and orientation, in a two-dimensional image. From these examples it can be seen that the alignment algorithm finds a small number of reasonable matches of widgets to images, even when the widget is forshortened, scaled, and partly occluded.

Acknowledgments

This paper describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the iaboratory's artificial intelligence research is provided in part by the System Development Foundation and in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N0014-85-K-0124.

References

1. Asada, H. and Brady, M. 1984. The Curvature Primal Sketch, MIT Artificial Intell. Lab. Memo, No. 758.

2. Baird, H. 1986. *Model-Based Image Matching Using Location.* Cambridge, Mass: MIT Press.

3. Besl, P.J. and Jain, R.C. 1985. Three-Dimensional Object Recognition, *ACM Computing Surveys* bf 17(1), pp. 75-154.

4. Bolles, R.C. and Cain, R.A. 1982. Recognizing and Locating Partially Visible Objects: The Local Feature Focus Method, *Int. J. Robotics Res.* **1**(3), pp. 57-82.

5. Bolles, R.C. and Horaud, P. 1986. 3DPO: A Three-Dimensional Part Orientation System *Int. J. Robotics Res.* **5**(3), pp. 3-26.

6. Brady, M. and Asada, H. 1984. Smoothed Local Symmetries and Their Implementation, MIT Artificial Intell. Lab. Memo, No. 757.

7. Brooks, R.A. 1981. Symbolic Reasoning Around 3-D Models and 2-D Images, *Artificial Intell. J.* **17**, pp. 285-348.

8. Canny, J. 1986. A Computational Approach to Edge Detection, *IEEE Trans. Pat. Anal. and Mach. Intel.* **8**(6), pp. 679-698.

9. Faugeras, O.D. and Hebert, M. 1986. The Representation, Recognition, and Locating of 3-D Objects, *Int. J. Robotics Res.* **5**(3), pp. 27-52.

10. Fischler, M.A. and Bolles, R.C. 1986. Perceptual Organization and Curve Partitioning, *IEEE Trans. Pat. Anal. and Mach. Intel.* **8**(1), pp. 100-104.

11. Foley, J.D. and VanDam A. 1984. *Fundamentals of Interactive Computer Graphics.* Reading, Mass: Addison.

12. Grimson, W.E.L. and Lozano-Pérez, T. 1984. Model-Based Recognition and Localization from Sparse Range or Tactile Data, *Int. J. Robotics Res.* **3(3)**, pp. 3-35.

13. Grimson, W.E.L. and Lozano-Pérez, T. 1987. Localizing Overlapping Parts by Searching the Interpretation Tree, to appear in *IEEE Trans. Pat. Anal. and Mach. Intel.*.

14. Hoffman, D.D. and Richards, W.A. 1986. Parts of Recognition, in *From Pixels to Predicates: Recent Advances in Computational and Robotic Vision*, edited by A.P. Pentland. Norwood, N.J.: Ablex.

15. Huttenlocher, D.P. and Ullman, S. 1987. Recognizing Rigid Objects by Aligning Them with an Image, MIT Artificial Intell. Lab. Memo, No. 937.

16. Lowe, D.G. 1985. *Perceptual Organization and Visual Recognition.* Boston: Kluwer.

17. Lowe, D.G. 1986. Three-Dimensional Object Recognition from a Single Two-Dimensional View, Courant Institute Robotics Report, No. 62.

18. Witkin, A.P. 1983. Scale-Space Filtering, in *From Pixels to Predicates: Recent Advances in Computational and Robotic Vision*, edited by A.P. Pentland, pp. 5-19. Norwood N.J.: Ablex.

19. Yuille, A.L. and Poggio, T. 1983. Scaling Theorems for Zero Crossings *IEEE Trans. Pat. Anal. and Mach. Intell.* **8(1)**, pp. 15-25.

Parallel Recognition of Objects Comprised of Pure Structure

Paul R. Cooper and Susan C. Hollbach

Department of Computer Science
University of Rochester
Rochester, New York 14627

Abstract

We are interested in the problem of object recognition from visual information. We address two novel and related subproblems in particular: the recognition of objects given a large number of possible alternates, and the recognition of objects consisting of a pure structural composition of primitives.

We examine these issues in the context of a mini world problem: the recognition of models of animals constructed with Tinker Toys. Initial results include the development of an end-to-end system that uses 2D connectivity to recognize views of Tinker Toy objects. Topological matching is performed in parallel by a connectionist network that propagates binary constraints.

1. Introduction

The central goal of vision (in both man and machine) is *perception*: to know what is out there, to recognize the things making up a visual environment. Much recent research has concentrated upon the many sub-issues that have arisen on the way to this goal, such as adequate reconstruction of shape. In contrast, we wish to again focus upon actually accomplishing recognition from input image data. Thus we adopt the mini-world approach to vision research [Mackworth 1978]. This entails building an end-to-end system that can interpret images of a limited domain, or mini-world.

In this paper, we propose a new mini-world problem: the recognition of models of animals constructed with Tinker Toys. The problem involves two central novelties. First, the objects to be recognized comprise a complex composition of very simple elements - essentially pure structure. The second novelty is addressing a large model base of alternate objects. Such a model base can be generated in a very constrained manner when objects are constructed by composition.

Following a description of these issues, we outline the basic framework of our solution and the parallel implementation we have constructed. In our experiments, we have focussed upon using domain dependent search constraints to eliminate candidate models from the model base. We describe in particular a connectionist implementation of binary constraint propagation that does topological matching of Tinker Toy objects to models. We observe that this parallel process is very effective in constraining the recognition search, even when it faces a moderately large model base.

2. The Tinker Toy World

2.1 What: The Problem

One advantage of the mini-world approach is that it simplifies the task of making a clear problem statement. Tinker Toys consist of two main primitives, rods of differing lengths (axes, loosely speaking) and disks (or joints), as seen in Figure 1. Tinker Toy objects are any connected objects composed from these primitives. In its most general form, the Tinker Toy

World consists of scenes of one or more of these objects. In this work we consider mainly scenes of only one object. The goal is to determine which of a set of object models best represents the real object in view.

2.2 Why: The Motivation

Two complementary characteristics of the Tinker Toy domain motivated its selection. First, it allows us to investigate the two novel aspects of the recognition problem that arise in the Tinker Toy world. Second, it minimizes the effort required to address any but these two subproblems.

a) Recognition from Structure

The first issue we explore with the Tinker Toy world is the role of structure in recognition [Witkin and Tenenbaum 1983]. We take structure to be a stratified and hierarchical description of the spatial relationships of different entities to each other. Thus, global scene structure is the relative position of the objects to each other. Object structure in turn is the spatial composition of the object with respect to its sub-components. Sub-components may themselves be similarly structured. Finally, structure in an image arises from the image projection of either scene, object, or component structure. Witkin and Tenenbaum [1983] observe that the most useful image structure is regularities that cannot be accounted for by accidents of projection and alignment. Typically, this is the image projection of basic component structure. On the other hand, the image projection of global scene structure can be used to do stereo matching [Cooper et al 1985, Cooper 1987].

Our concept of structure as embodied by Tinker Toy objects thus corresponds to a theory of shape representation by parts and composition, reduced to its simplest essentials. Pentland [1986], Biederman [1985], Hoffman and Richards [1985], and Marr and Nishihara[1983] all advocate shape representation by parts and composition. This theory requires two distinct components - a description of the spatial relationships between the parts and how they compose to form the whole (the structural aspect), and an in-depth description of the parts themselves (largely a description of the surfaces of the parts). (Alternatives exist. For example, Michael Leyton[1986] suggests that some kinds of shapes might be best represented and analyzed in terms of formative growth processes.)

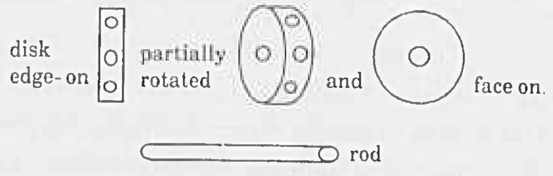


Figure 1: Tinker toy primitives

The Tinker Toy world makes an important simplification from the general theory of shape representation by parts and composition. That is, the primitive parts (the rods) are constrained to be both straight and lacking in any significant surface variation. In effect, *surfaces are irrelevant in the characterization of Tinker Toy objects*. This constraint accomplishes the following. First, it allows us to perform recognition from information that almost purely characterizes the structural aspects of shape (as opposed to the surface aspects). Second, it allows us to avoid directly addressing the difficult problems of reconstructing and representing (with invariance!) surface shape. Finally, it simplifies the low-level processing required to actually construct an end-to-end recognition system.

For most recognition schemes, the parts (and the shapes of the parts themselves) are critical to the theory (e.g. in the recognition-by-components theory of Biederman [1985]). In the Tinker Toy world, the parts offer virtually no information for recognition. It is only the non-local structure of the object that can be used as a basis for recognition. This same structural information must ultimately be used to verify perception, even in recognition-by-component theories.

b) Recognition from a Large Model Base

Allowing objects to comprise composed structure implicitly defines a potentially very large number of objects that could be recognized. This is the second major facet of the recognition problem that the Tinker Toy domain allows us to focus upon.

In the Tinker Toy world, the large model base is generated in a very constrained fashion (by composition *only*). As a result, we need not consider distracting issues such as the selection of the most relevant feature (for recognition of objects with color, texture, curved shape, motion etc). Instead, we must determine how to distinguish an object from many others based only on its structure.

2.3 Other Miniworlds

This domain and task contrast sharply with reconstruction oriented research, such as the 'shape from *x*' work. But the Tinker Toy world differs as well from previous mini-world research. Blocks world was the original mini-world [Roberts 1965, Huffman 1971, Clowes 1971, Waltz 1975]. In this world, the surface shape of the objects was clearly critical, although some composition was allowed. Polyhedral world [Ballard 1986] depends upon a boundary characterization of surfaces, as does Airplane world [Brooks 1981]. A much different mini-world under recent investigation is the Sketchmap world of Mackworth [1978], Mulder [1985] and others. An important aspect of this mini-world is the provision for (and crisp distinction between) composition and categorization hierarchies in the domain knowledge. In Tinker Toy world, the composition hierarchy is obvious; the categorization hierarchy arises from constraining the objects to be models of animals. Even so, Tinker Toy world differs from the Sketchmap world because it deals with 3D objects instead of 2D sketchmaps.

In summary, we see that the Tinker Toy world is a domain that allows us to focus upon the role of spatial structure in the recognition of real, hierarchically composed 3D objects from a large model base. The domain also allows us to largely ignore low-level vision issues, as well as other problems such as surface representation and recognition.

3. Solution Framework

3.1 Cycle of Perception

We adopt the hypothesize-and-verify view of visual recognition [Roberts 1965], which Mackworth[1978] aptly calls the Cycle of Perception. We take the hypothesize phase to primarily involve the solving of two subproblems: searching the space of objects and searching the space of viewpoints. Following this, the verification phase establishes a correspondence between the hypothesized object and hypothesized viewpoint, and invariants registered to the image data. This partitioning of the recognition problem corresponds intuitively (in the worst case) to generating images of every single object in the model base from all possible viewpoints, then comparing these images to the input imagery until a match is found.

We adopt two basic principles in addition to the central Cycle of Perception framework. The first is the use of principal views to represent qualitative viewpoint invariance. The second is the use of domain dependent criteria (developed with some consideration as to efficiency) to prune the search space of possible objects. We expand slightly on each of these ideas in what follows.

3.2 Principal Views

Much previous work in visual recognition has focussed upon determining viewpoint - i.e. computing the coordinate frame transformation between the object as modelled and its instantiation in the world, via the intermediate image coordinate frame (e.g. [Ballard 1986]). This requires a good hypothesis as to the identity of the object. In model-based systems where there are a small number of objects or the object's identity is known, this is a reasonable approach. In general however, there is a large space of modelled objects that must be searched also, as in the Tinker Toy domain. A chicken-and-egg dilemma arises if the space of objects is to be searched first, as this is difficult without some idea as to the viewpoint.

The space of viewpoints is inherently quantitative and infinite. However, an object's qualitative appearance is usually invariant over a wide range of viewpoints. We capture this invariance by representing principal views of the objects in the model base[Feldman 1985, Chakravarty 1982, and Burns and Kitchen 1986]. In effect, principal views precompute part of the solution to the viewpoint determination problem, allowing us to bootstrap directly into searching the space of objects (bypassing the chicken-and-egg dilemma). Figure 2 presents four of the principal views of a Tinker Toy horse, along with the relational structures built from these principal views.

3.3 Search Constraint

The large model base still remains to be searched. From the model base, we must select a small number of models as reasonable hypotheses as to the identity of the object in view. Alternately, we must discard many bad candidate models.

Our response to this problem derives from two observations. First, we must develop domain dependent recognition criteria. As Mackworth[1978] notes, "only by patiently teasing out the semantics of a domain (that is, the relationship between objects in the world and their pictorial traces) will we be able to write programs which interpret pictures in that domain". The domain dependent criteria developed for the Tinker Toy world will have utility in general.

Our second observation is that computational cost is important. There are a combinatorially large number of distinguishable ways of composing very few Tinker Toy primitives. Furthermore, the cost of computing the distinction is in some cases enormous.

As a result, our approach is to develop domain dependent criteria to prune the space of objects, and to apply them in the most cost effective manner. In effect, we try to discard as many possible candidates as

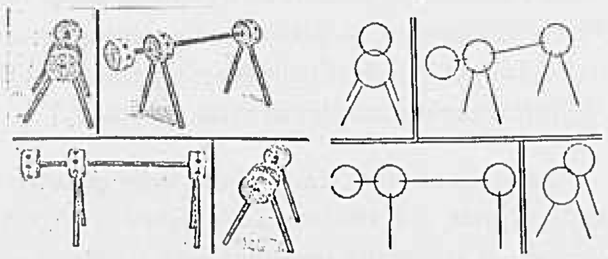


Figure 2: some principal views of a Tinker Toy horse

cheaply as possible. The idea is to begin filtering candidates with locally computable and computationally cheap tests, such as counts of the primitives and local junction analysis. Progressively more comprehensive and more costly tests (such as those involving non-local and especially geometric characteristics) are applied until sufficiently few hypotheses remain. When only a few good hypotheses remain, the verification phase of the Cycle of Perception can then be used to make final decisions, and iterate if necessary.

4. Experiments

4.1 Parallel Qualitative Discrimination

In our initial experiments, we have explored only a subset of the whole Tinker Toy interpretation problem. While we have indeed constructed a complete end-to-end system, the experiments have been developed to investigate two specific issues.

The first issue we explored was using discrimination criteria based solely on the *qualitative* characteristics of the objects to prune the model base. In particular, we have concentrated upon extracting and using only the topological characteristics of the objects in 2D views. In the Tinker Toy domain, this amounts to connectivity information - what is connected to what. Figure 3 shows some qualitatively or topologically similar objects, which differ in their geometric characteristics. In the higher level representations of the objects, essentially all quantitative geometric information is being ignored.

Because we have not been concerned with computing quantitative and geometric information, we have not performed a detailed computation of viewpoint. Therefore, metrics such as rod lengths are unavailable for discrimination or recognition. As well, we have constructed the principal views in the model

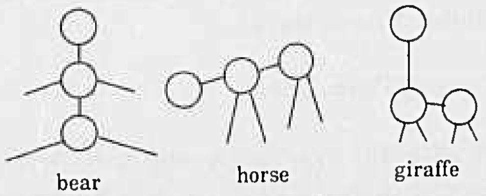


Fig .3:Three topologically identical figures

base ourselves, rather than generate them. Finally, we have not yet implemented the verification phase (which is quantitative in nature).

The second issue we addressed in our experiments was a massively parallel (connectionist) implementation of the key components [Feldman and Ballard 1982, Feldman 1986b]. This development was primarily motivated by the need to consider the potentially large database.

4.2 Implementation

The end-to-end Tinker Toy object recognition system has been implemented and tested on model bases of up to 18 objects, and is also described in [Hollbach 1986]. There are four stages: 2D primitive acquisition, 3D primitive acquisition, instance building and model invocation.

The 2D primitive acquisition stage takes camera output, optionally calibrates it, and runs a Kirsch operator over it to get gradient magnitude and direction information. The edges are then thinned by a simple one-pass thinning algorithm that strengthens locally maximal edges, ignores ambiguous edges and removes all others. For an edge to be locally maximal, the edges "in front of" and "behind" it (where an edge is deemed to be facing in the direction of its gradient) if pointing in the same direction must have smaller magnitude, and the edges to either side must both point in the same direction. An edge is ambiguous if it would be locally maximal but for the fact that only one neighbour points in the same direction. Both the edge detection and the edge thinning stages are highly parallel processes, since the computation at each edge depends only on the prior values of edges in the immediate neighbourhood.

The thinned edges are then segmented into line segments and circles via the Hough transform [Ballard & Brown 1982]. The Hough transform is a highly parallel process for recovering non-local image structure [Ballard 1984]. The local computations for line segment detection involve detecting a triple of collinear edge points, a slight modification of the standard method which uses only pairs. In a further modification, the endpoints of the line segment defined by these triples are collated by the voting process to yield an estimate of the endpoints of each line segment in the image. This obviates the need to search separately for endpoints along each line found in the image.

The 3D primitive acquisition stage is simplified by the fact that Tinker Toys only have two components, rods and wheels. Rods are built by pairing line segments of opposite gradient directions. Spurious matches between parallel rods are discarded by enforcing the condition that all rods have roughly the same width, which is taken to be the mode of all possible widths. Wheels are roughly located by clumping together all mutually intersecting circles. Although in general there are no true circles present in the image, this process works because the ellipses present in the image can be approximated by piecewise assemblies of circular arcs. The areas of circular activity in the image are highly localized, and correspond to the presence of the elliptical projection of a wheel.

Instance building is the process of creating a precise symbolic representation of the Tinker Toy from the low level data structures. A potential problem is created by the fact that the rods and disks may not actually intersect one another in the image. Thus a more sophisticated strategy than merely detecting intersections of circles and line segments is required. The strategy adopted is to create a quad tree spatial location array for the image such that each cell in the quad tree is occupied by only one disk. The structure is then traversed, and all rods present in each cell are deemed to be attached to the resident wheel, unless both ends of the rod occupy the same cell, in which case only the end closest to the wheel's center is attached. The resulting data structure carries both geometric and topological information.

4.2.1 Model Invocation by Connectionist Matching

In the model invocation stage, a parallel connectionist constraint propagation network running on the Rochester Connectionist Simulator [Fanty 86] is used to rank possible matches by topological similarity

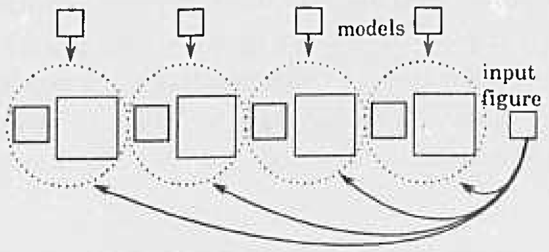


Figure 4: overall network structure

to the target figure. This process uses domain dependent unary and binary constraints to discard many candidate matches from the model base.

The basic idea is to match the input to all models in parallel, taking the highest scoring match to be the winner. Separate subnets exist for each complete potential figure-model match, as shown in Figure 4. Each subnet consists of two mutually interconnected "arrays" of units. The smaller left-hand array in each subnet represents all possible matches between disks in the figure and disks in the model. The larger right-hand array represents the possible matches between pairs of connected disks in the figure and pairs of connected disks in the model. Spreading activation along the connections between the two arrays ultimately computes the best match by constraint propagation.

In principle, the network encoding the model base and matcher remains intact for different input figures. Particular input instances are represented by which units are activated.

(i) Network Configuration

The detailed architecture of the subnets is described in Figures 5 and 6. Figure 5 shows a network for

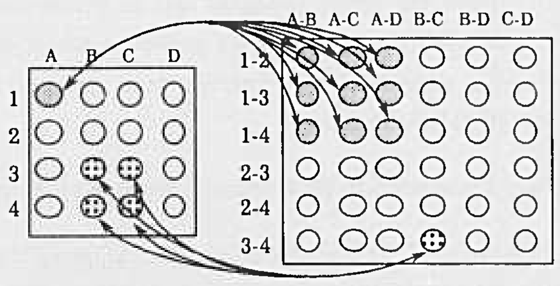


Figure 5: wheel matching array, constraint matching array, and example constraint propagation links

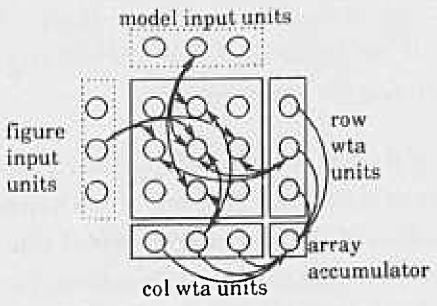


Figure 6: detail of array structure

matching an image-derived figure with four disks against one candidate model, also with four disks. In this example, figure disks are assigned arbitrary numeric labels, and the disks of the model figure are labelled with letters.

In the disk matching array (Fig. 5, left), rows correspond to figure disks, and columns to model disks. Thus, the *ij*th entry in this array corresponds to a match between the *i*th figure disk and the *j*th model disk. Units in the disk matching array are activated if the unary constraints on the corresponding disks are equal. In these experiments, the unary constraint at a disk is the number of rods emanating from that disk.

In the constraint matching array (Fig. 5, right), each row and column corresponds to a potentially connected pair of disks. The *ij*th entry in this array thus represents a match between the *i*th binary constraint of the figure, and the *j*th binary constraint of the model. Not all potential pairwise connections are actually present in the figure or model. The units active in the constraint matching array correspond to matches between connected pairs of disks that are actually instantiated in the figure and model.

The connections between the arrays reinforce mutually agreeable unary and binary matches. Each unit in the constraint matching array is bidirectionally linked to four units in the disk matching array. These four units correspond to the four possible disk matches which reinforce that constraint match. For example, in Figure 5, the constraint unit at the intersection of the "3-4" row and the "B-C" column attaches to the wheel units at the intersections of the "3" and "4" rows and the "B" and "C" columns.

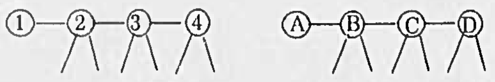


Figure 7: example of a figure and matching model

There are additional overhead units and connections whose job it is to extract the answer. Each row and column in both networks is connected in a winner-take-all network, as shown in Figure 6. Other units accumulate and compare subnet totals.

(ii) Network Operation

The network operates synchronously. Each step consists of two substeps, the winner-take-all contest, and the propagation of constraints between the unary and binary constraint matching networks. In the winner-take-all, units which win in both their row and column are reinforced, to create a 'match' binding. All other units (which either lost or tied in a winner-take-all) become minimally active (e.g. potential of 1). In the constraint propagation phase, each unit computes the sum of its inputs, from the other half of the network. Bound 'winners' have high potential, and propagate back and forth, reinforcing further consistent matches.

(iii) Example

Consider the example figure and model in Figure 7, and the corresponding initial network configuration in Figure 8. The figure consists of 4 disks connected linearly by 3 rods, plus additional "dangling" rods. There are $n(n\text{-}1)/2 = 6$ possible connections between the 4 labelled disks of the figure: 1-2, 1-3, 1-4, 2-3, 2-4, and 3-4. In this particular instance only 3 of the possible connections are realized, 1-2 , 2-3, and 3-4. The model is labelled correspondingly with letters.

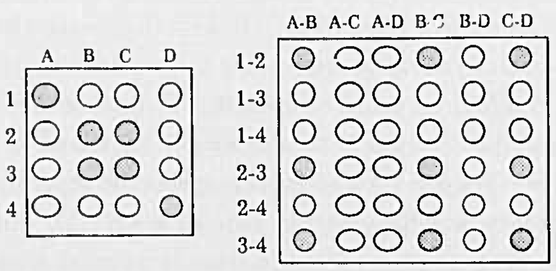


Figure 8: possible wheel matches are A1, B2, B3, C2, C3 and D4.

In the disk matching array, the units corresponding to a match of disks with an equal number of rods attached are all active. Only the '1' disk and the 'A' disk have a single attached rod, for example. In other cases, more than one match is possible from the unary constraints, and units are activated to reflect that fact.

In the constraint matching array, the units are on at the intersection of rows and columns coresponding to instantiated connections. In this network, all active units reflect all potential matches of pairs of disks.

In this example, the unit at 'A1' is uniquely active in its row and column, which reflects the fact that there is a unique binding of disk-1-rod labels. In the first winner-take-all, this unit wins. After iteration, this binding propagates, reinforcing the '12-AB' pair match on the first step. Eventually, the subnets settle, and the best overall match for this pairing is selected. Simultaneously, all other figure-model pairings are being compared, and the network selects the best match overall.

4.3 Results

The system's operation in one experiment is summarized in Figures 9 and 10, and Table 1. The results of additional tests of the matching network are described in Table 2.

Figure 9 describes the operation of the early stages of the system. Figure 9a) shows a digitized picture of a Tinker Toy horse, the input to the system. Figure 9b) represents the magnitude of the gradient produced by the Kirsch operator (ie. the edge picture). Figure 9c) has both straight lines and circles found by the Hough

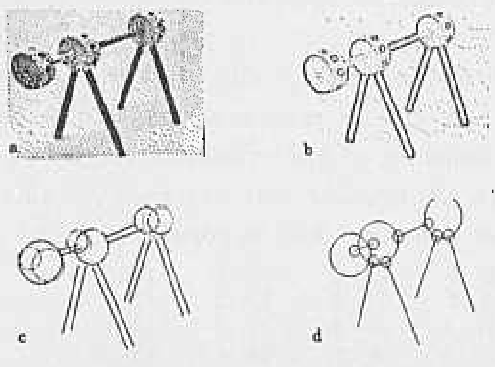


Figure 9: lower level processing

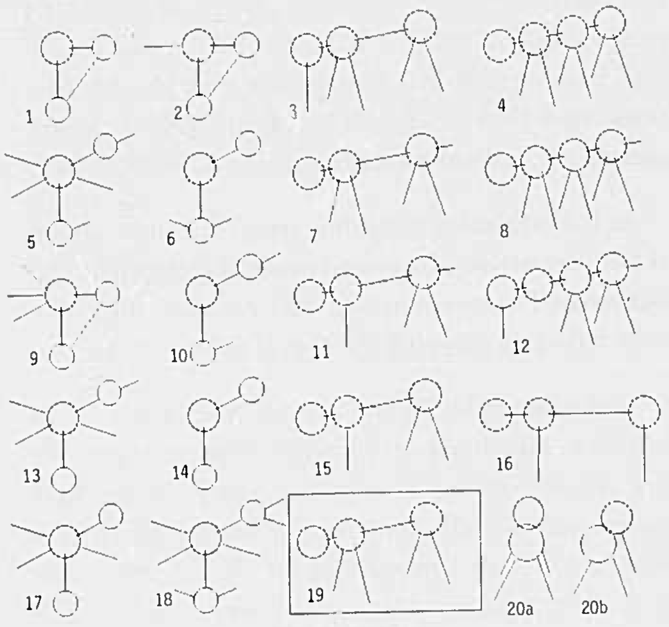


Figure 10: the model base

technique from the edge information of 9b). And 9d) demonstrates the connectivities found between rods and disks: the small circles represent joints between a rod and disk.

Figure 10 shows the model base of 18 objects, plus three additional views of the 'horse' object. The connectionist matching network simultaneously compares each topological principal view in the model base with the extracted topological representation of the actual object shown in Figure 9. The results of the matching are tabulated in Table 1. Scores are normalized to the value of a perfect match, so that the match of an object to itself rates 1.0. We see the central goal of recognizing the target 'horse' figure is accomplished, as its similarity score is the highest. Also of note are the reasonable scores obtained by partial matching of approximately similar figures. Although the network was designed primarily to accept or reject matches of objects with the same number of disks, partial matches and matches of objects with differing numbers of disks are computed also. For example, compare the extracted figure with models 3 and 7 (judged quite similar) and models 4, 6 and 15 (judged fairly similar).

1) 0	6) .4	11) .2	16) .2
2) 0	7) .6	12) .2	17) 0
3) .6	8) .2	13) .2	18) 0
4) .4	9) 0	14) 0	19) 1.0
5) 0	10) 0	15) .4	20) 0

Table 1: match scores against figure 19 (the horse)

Figure 9

1) 0	6) 0	11) 0	16) 0
2) 0	7) 0	12) 0	17) .5
3) 0	8) 0	13) .5	18) .5
4) 0	9) 1.0	14) 0	19) 0
5) .5	10) 0	15) 0	20) 0

Figure 11

1) 0	6) .2	11) 1.0	16) .6
2) .18	7) .4	12) .2	17) 0
3) 0	8) .4	13) .2	18) 0
4) .2	9) 0	14) 0	19) .2
5) 0	10) 0	15) .6	20) 0

Table 2: match scores against figures 9 and 11

4.4 Matcher Analysis

The network whose performance we have just described essentially computes a similarity metric between labelled graphs. It does not represent an attempt to solve the graph or sub-graph isomorphism problem. Instead, it is an inherently resource-bounded process intended to discard as many candidate models as is feasible within the resource constraint. The network is similar in principle to work such as Kitchen and Rosenfeld's [1978] study of matching relational structures by relaxation. It is also similar to the work of Ambler et al [1975] on graph matching by clique finding. It differs in motivation and implementation detail in a variety of ways, however.

Complexity

One notable difference is the fact that the parallelism in our network is explicitly represented. This makes extending the network in parallel for large numbers of objects very easy to do. More importantly, the explicit parallelism facilitates a complexity analysis of the algorithm. A key characteristic of connectionist networks which determines their viability is their space complexity. In this case, if k is the number of objects in the model base and n the number of disks in a Tinker Toy object (and model), the constraint matching arrays require $k(\frac{1}{2}n(n\text{-}1))^2$ units: $O(n^4)$. The number of connections is at most a small constant times the number of units. Typically, n will be less than 10. (Otherwise, a hierarchy suggests itself, but that is outside the scope of this discussion). The network which we constructed to perform the experiment had $k=21$, $n=4$, so the total number of units used was about 2100. For a model base containing 20 objects with 10 disks each, about 40,000 units would be required. In general, this seems an entirely reasonable requirement. For example, it is a requirement quite easily met by our own simulator.

On the other hand, it is fairly clear that the network is near the acceptable upper bound of complexity. This suggests that exhaustively computing more expensive characterizations of the graphs might not be affordable.

As with most parallel algorithms, an advantage of this implementation is its time complexity. While we present no formal proofs here, it seems fairly clear that the worst case running time is linear in the number of disks in a Tinker Toy object. Similarly, it is obvious enough that the network will settle, as it possesses no mechanism which could ever allow it to diverge.

Flexibility

A much more interesting result of this parallel approach is the flexibility it yields. The algorithm implemented in the network performs an entirely local analysis of the unary and binary constraints embodied in the Tinker Toy graph. By design, it is guaranteed to find and match any unique unary or binary constraints. Then, it propagates those matches as far as consistently possible, given limitations due to local symmetries which are not detectable by local analysis. Of couse, all this occurs in parallel.

The result, as the experimental results reveal, is surprisingly robust. In effect, the maximum number and size of easily recognizable correspondences (subgraph matches) are quickly and cheaply detected. In this case, 'easily' amounts to 'discriminable by local unary and binary analysis'. The result is a reasonable ranking of topological similarity, which is computable even for graphs of differing sizes. Such flexibility is indispensable when dealing with many complications which can arise in real vision problems.

5. Conclusions

This paper has examined the problem of recognizing objects comprised of pure structure, given a large model base of potential alternates. The lack of part or surface features to use for indexing suggests that a costly non-local representation of structure must be used to discriminate between them. But the large model base suggests the need for efficient indexing criteria.

We have described experiments with a complete end-to-end system which demonstrate the existence of a useful middle ground. Matching and propagating unary and binary constraints was found to be both affordably computable and surprisingly effective at discarding candidate models. In many situations, recognizing one composed object from many alternates requires discrimination based on structure. Our paper demonstrates that one can discard many potential candidates without resorting to quantitative and geometric measurement, and without explicitly computing non-local characteristics of the object. In certain circumstances, a local analysis of qualitative characteristics of an object is sufficient to successfully identify it.

It is also worth noting that computing such local qualitative criteria requires only modest performance from low-level processing. As a result, the criteria may be reliably computed from real images, yielding an end-to-end system successful in its domain.

An important aspect of our solution was its parallel character. Our connectionist implementation considered all unary and binary constraint matches to all models in parallel. A complexity analysis of this explicitly parallel implementation shows it to require polynomial space and linear time steps, acceptable even for large numbers of objects. Beyond this, the connectionist implementation was seen to demonstrate useful flexibility in ranking even inexact Tinker Toy matches by topological similarity.

In our future work, we plan to explore what geometric criteria to compute for more discriminating recognition within the Tinker Toy animal domain. We also plan to consider multiple objects in a scene, as well as problems arising from attempting recognition with uncertain and inexact information.

Acknowledgements

Nigel Goddard was instrumental in the design of the connectionist matching network. Jerry Feldman was an indispensable help throughout. Discussions with Mark Fanty and Mike Swain contributed.

This work was supported in part by NSF Coordinated Experimental Research grant no. DCR-8320136, in part by NIH Public Health Service research grant no. 1 R01 NS222407-01, and in part by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Foce Base, New York 13441-5700 and the Air Force Office of Scientific Research, Bolling AFB, DC 20332 under Contract No. F30602-85-C-0008. The latter contract supports the Northeast Artificial Intelligence Consortium (NAIC). We thank the Xerox Corporation University Grants Program for providing equipment used in the preparation of this paper.

References

Ambler, A.P., H.G. Barrow, C.M. Brown, R.M. Burstall, and R.J. Popplestone. "A versatile computer-controlled assembly system", *Artificial Intelligence 6*, 2, 129-156, 1975.

Ballard, D. "Form perception using transformation networks: Polyhedra," TR 148, Computer Science Department, University of Rochester, October 1986.

Ballard, D. "Parameters Nets", *Artificial Intelligence 22*, 235-267, 1984.

Ballard, D. and C. Brown. *Computer Vision*, Englewood Cliffs, N.J.: Prentice Hall, 1982.

Biederman, I. "Human image understanding: recent research and a theory", *Computer Vision, Graphics and Image Processing* Vol. 32, No. 1, pp. 29-73, 1985.

Brooks R.A. "Symbolic Reasoning among 3D models and 2D Images", *Artificial Intelligence*, vol. 17, pp. 285-345, August 1981.

Burns, J. Brian and Les Kitchen. Personal Communication, May 1986.

Clowes, M.B. "On seeing things", *Artificial Intelligence 2*, 1, pp.79-116, Spring 1971.

Chakravarty, I. and H. Freeman. "Characteristic Views as a basis for three dimensional object recognition", SPIE Vol. 336 (Robot Vision), 37-45, 1982.

Cooper, Paul R., Daniel E. Friedmann and Scott A. Wood. "The Automatic Generation of Digital Terrain Models from Satellite Images by Stereo". 36th Congress of the International Astronautical Federation. Stockholm, Sweden, October, 1985. To appear in Acta Astronautica, the Journal of the International Academy of Astronautics.

Cooper, Paul R. "Order and Structure in Correspondence by Dynamic Programming". To appear, 1987.

Fanty, M. and N. Goddard."Users manual, Rochester connectionist simulator", TR in preparation, Computer Science Department, University of Rochester, 1986.

Feldman, Jerome A. "Four Frames Suffice: A provisionary model of vision and space." *The Brain and Behavioural Sciences 8*, 265-289, 1985a.

Feldman, Jerome A. "Connectionist Models and Parallelism in High Level Vision", *Computer Vision, Graphics, and Image Processing 31*, 178-200, 1985b.

Feldman, J.A. and D. Ballard. "Connectionist models and their properties," *Cognitive Science*, 6, 205-254, 1982.

Hoffman, D. and W. Richards, "Parts of recognition" in *From Pixels to Predicates*, Pentland, A., (Ed.) Norwood, N.J.: Ablex Publishing Co. 1985.

Hollbach, S.C. "Tinker Toy recognition from 2D connectivity", TR 196, Computer Science Department, University of Rochester, October 1986.

Huffman, D.A. "Impossible Objects as nonsense sentences". In *Machine Intelligence 6*, 1971.

Kitchen, Les and A. Rosenfeld. "Discrete Relaxation for Matching Relational Structures", *IEEE Trans. Systems, Man, and Cybernetics, SMC-9, 12*, 869-874, 1979.

Lowe, David. *Perceptual Organization and Visual Recognition*, Boston MA: Kluwer Academic Publishers 1985.

Leyton, Michael. "A Process Grammar for Shape". To appear.

Mackworth, A.K. "Vision Research Strategy: Black Magic, Metaphors, Mechanisms, Miniworlds and Maps", in *Computer Vision Systems*, A. Hanson and E. Riseman, (Eds.). New York: Academic Press, 1978.

Marr, D. and K. Nishihara. "Representation and recognition of the spatial organization of three-dimensional shapes" in *Vision*, David Marr. San Francisco: W.H. Freeman and Company, 1982.

Mulder, Jan A. Using Discrimination Graphs to Represent Visual Knowledge. TR 85-14, Laboratory for Computational Vision, Department of Computer Science, University of British Columbia, 1985.

Pentland, A.P. "Parts: Structured Descriptions of Shape", *Proceedings, AAAI-86*, 695-701, Phil., PA, Aug. 11-15, 1986

Roberts, L.G. "Machine perception of three-dimensional solids" in *Optical and Electro-optical Information Processing*, J.P. Tippett et al. (Eds.). Cambridge, MA: MIT Press, 1965.

Waltz, D.I. "Generating semantic descriptions from drawings of scenes with shadows." in *The Psychology of Computer Vision*, P.H. Winston (Ed.). New York: McGraw Hill, 1975.

Witkin, A.P. and J.M. Tenenbaum, "On the role of structure in vision", in *Human and Machine Vision*, J. Beck, and A. Rosenfeld, (Eds.). New York: Academic Press, 1983.

A FRAMEWORK FOR IMPLEMENTING MULTI-SENSOR ROBOTIC TASKS

Peter K. Allen

Department of Computer Science
Columbia University
New York, NY 10027

Abstract

Complex robotics tasks such as assembly and manipulation require the use of multiple, powerful and accurate sensing devices. Unfortunately, design of a robotics system usually entails a number of sensor specific, task specific, ad-hoc modules that relate low level sensing to the task at hand. This structure make it difficult to upgrade the system if new sensors or sensor strategies are developed. In addition, changing the system to perform a related or similar task may require a large effort. A potential solution to this problem is to organize sensors and sensory-related robotic tasks as classes and methods in an object-oriented programming system. Objects are represented by *classes* which can partition sensors and tasks by attributes and behaviors, and *methods* which allows specialized procedures, local to a class, to be invoked. This allows similar sensing behavior to take place in spite of possibly different underlying physical sensors. This paper discusses the implementation of two proposed logical, high level virtual sensors: an intelligent hand for grasping and a mobile eye that can perform motion tracking. The former contains tactile sensors and finger servos to control coordinated grasping and tactile recognition and the latter combines the sensing capability of vision methods with a manipulator arm class upon which the camera is mounted.

1. INTRODUCTION

The problem of multi-sensor integration for complex robotic tasks is a difficult one. In some ways, it is a metaphor for many areas of current Computer Science research, including distributed systems, real-time computing, signal processing, knowledge representation and high level artificial intelligence methods. In trying to build multi-sensor systems, one can quickly become bogged down in the details of physically integrating devices, low level sensor processing, error recovery and the like, preventing a focus on the larger issues of building robust, modular, expandable systems. The resulting systems tend to have an ad-hoc, task and sensor specific flavor.

What is needed for significant progress in this area is a programming paradigm and framework for building such systems. Achieving this has been difficult for a number of reasons. First, the tendency has been to work on parts of the problem separately (they are challenging enough by themselves!) without an overall system view. Secondly, the sensor systems are usually self contained processing systems, not easily integrated into a larger system. They rarely possess sophisticated programming environments that allow for the creation of robust high level software systems. Lastly, the requirements of an integrated sensing system have been ill defined. The payoff in building a framework for sensor integration is twofold. One, the system is maintainable and modifiable as new and better sensing devices come along, and secondly, it establishes an experimental environment that facilitates the creation of new strategies and methods of integration. We do not foresee any systems in the near future that will be intelligent enough to generate their own strategies given a set of sensors and a task description (although it remains an interesting research topic). Therefore, the expertise of the programmer is needed to generate sensing strategies, operators and integration methods. As usual, the burden is on the programmer. By developing a framework such as the one described here, the programmers burden will be lessened, allowing more experimentation and faster progress.

Previous work in this area has been done by Henderson and his group at the University of Utah [9]. The focus of their research was in building a logical sensor specification (LSS) that would allow logical or virtual devices to be specified for a task. These devices were abstractions of sensor data that was available from (possibly) many sources and in many formats. The logical sensor module would interface the higher level programming system with the underlying physical implementations of the sensing devices. This would be accomplished by outputting a characteristic vector of attributes and values for a logical sensor, regardless of underlying physical implementation. This paper is an elaboration of these ideas and includes a framework for a specific set of common robotic sensing devices and generic tasks.

As mentioned previously, there are a number of ancillary issues involved in designing such systems. One is the relationship between a high level knowledge based sensory system and the real-time demands of robotic computation. We will not address this issue directly in this paper, but note the work of Korein et al [13] at IBM in trying to build a comprehensive knowledge based system in concert with real-time actuation of robotic devices. Another issue that is important in this scope is concurrency and parallel operation which seem to be indispensable in active sensory systems involving many devices and processors. Work by Chiu et al [5] has attempted to deal with this issue. Our focus is

Acknowledgement: This work was supported in part by: NSF grant DMC-86-05065 and DARPA grant N00039-84-C-0165

on the overall software environment for an active sensor system incorporating many sensors (vision, touch, range, force/torque) and high level knowledge about a task and domain. The work here establishes an important experimental framework that will facilitate experimentation and progress in these other areas.

2. RELATING HIGH LEVEL TASKS TO SENSING

Brady has defined robotics to be control based on intelligent interpretation of sensor data in terms of a task plan and model [3]. This involves merging low level, sensor dependent modules that perform primitive functions with high level task descriptions. Defining this relation in a programming environment has proved difficult. One approach, advocated by Rodney Brooks at M.I.T. [4], is to develop a layered system based upon functional behaviors. In this model, increasingly complex behavior is able to subsume less complex behavior, allowing for graceful and controllable increases in a system's capabilities. While this model appears promising for the lower sensing and actuation levels, it may have problems as task level behavior becomes complex and strict hierarchical subsumption of one module by another fails. What appears to be a low level task in one scenario may be a high level task in another domain.

An alternative to a hierarchical system of layered behavior is a set of completely autonomous agents with arbitrary control function wired into them, commonly referred to as a blackboard model [7, 11]. In this model, global knowledge is posted to a common database, and the values in this database provide the control of the systems functions. Both blackboards and layers have been used to build mobile robots, and it is not clear which paradigm is superior. This paper advocates an approach to building these systems that is compatible with either of these approaches. The central idea is to relate sensing to actuation by treating these entities as objects in an object-oriented programming system.

3. OBJECT-ORIENTED PROGRAMMING

Object-oriented programming is an emerging paradigm in software engineering that tries to treat programs as entities consisting of objects (both data and procedures) that are to be manipulated. It embraces a number of important ideas that set it apart from the more traditional programming paradigms of procedures and accompanying data structures. Among them are:

- **Encapsulation:** Objects consist of *classes* and *methods* that encapsulate computations on these objects. Classes are ways of organizing similar data objects that will be acted upon in a similar way. Methods define procedures to be used on the objects defined in a class. These methods usually are private to the object class, meaning there are no side effects on objects that are not members of the class.
- **Data Abstraction:** Calling programs need not know or be concerned with the internal representations of data objects they wish to use. This abstraction is effected by making the unit of manipulation an object which has its own internal definitions and data structures. This has the added benefit of allowing changes to data objects to be transparent to client users of the objects. This is especially important in implementing logical sensors that are defined by different underlying physical devices.
- **Polymorphism:** Client users of objects can refer to them in many ways. Operator overloading is a common way of implementing this in such systems. It allows an object to respond to requests that have possibly different data representations. An example of this would be computing a position in space in either spherical, cylindrical or Cartesian coordinates.
- **Inheritance:** Similar objects can inherit properties (methods and data values) that are a natural result of a hierarchical description of the world. For example, all tactile sensors can belong to a class that defines displacements due to touch. Each individual sensor can then be further described as its own object, but without redundantly supplying class information that can be assumed by a more specialized object.

A useful way of thinking of object-oriented systems is as a set of entities (objects) that communicate arbitrarily via messages that cause object methods to be invoked depending upon the message content. Thus, this paradigm can emulate hierarchical behavior as well as more autonomous networks with arbitrary connections. There are a number of object-oriented languages in use today and their number is growing. Representative of such languages are Smalltalk-80 and CommonLoops. These and other object-oriented languages are reviewed in an excellent survey by Stefik and Bobrow [12] .

4. OBJECTS FOR MULTI-SENSOR ROBOTIC TASKS

The partitioning of classes into particular instances of that class maps directly onto the idea of a generic, logical sensor, having common sensing behavior but perhaps different physical implementation. In addition, the idea of inheritance is expressed in each class instance retaining default and acquired attribute values from an object class. Thus, a generic vision sensor can impart default attributes (resolution, gray levels) onto a particular instance of that sensor, which is easily changed if a new and different sensor is employed in the system. A generic tactile sensor reporting displacements and surface normals can be implemented as a single class with multiple instances in spite of the underlying grid structure or sensing medium.

The structuring of objects proceeds on a hierarchical basis. At the top level are task related classes with integrated methods that access more than one kind of sensor class. Underlying these integrated methods are sensor specific methods that are abstracted from generic sensing tasks into low level, sensor specific methods, culminating in each physical device, which is modeled with its own specific attributes.

To demonstrate these ideas, we are currently building a set of objects that define two high level tasks that utilize more than one kind of sensor. The tasks are identifying, picking up and manipulating an industrial part, a task we define as an *intelligent hand* and a system that tracks a moving industrial part in a workspace which we define as a *mobile eye*. Both of these tasks are being implemented in an experimental domain that consists of a 6-DOF PUMA robotic manipulator, a wrist mounted camera, and an intelligent tactile sensor/parallel jaw gripper mounted on the PUMA (figure 1). The gripper, manufactured by LORD corporation, contains two independent tactile/vector sensors mounted on independently servoed fingers (figure 2). The tactile sensors are 10x16 matrix arrays of sensing elements mounted on a strain guage block that can compute orientation vectors and torques on the fingers.

The intelligent hand object integrates the low level sensory capability of the tactile sensors on the fingers, the grasping capabilities of the fingers, and the movement of the hand on the arm to which it is attached. Such a hand should be capable of the following two functional behaviors: identify a part through tactile sensing and locate, grasp and pick up the part.

The mobile eye virtual sensor combines the sensing capability of vision methods with a manipulator arm class (which can be independent of a particular manipulator configuration: SCARA, Cartesian or revolute) upon which the camera is mounted. This eye should be capable of tracking a known moving part.

4.1. IDENTIFY PART

The first task that the intelligent hand should be capable of is to identify a part to be grasped. We will create an object class called **identify_part,** shown in figure 3, which will have methods associated with it to accomplish this task. The methods that **identify_part** will use are the following:

- **probe_part_method:** This method will call upon the classes **find_surface, find_hole, and find_cavity** to do feature based recognition on the part. Our previous work in integrating vision and touch [1] confirmed that these features are particularly useful in recognition tasks involving touch. Each of these classes contain methods that will find the appropriate feature or return a message saying they were not able to find the feature. The probing of the environment to find the features requires a a sensing strategy that uses both vision classes and touch classes. New strategies, involving other sensors or features, may be added as another method in this class.
- **match_part_method:** This method will take a feature found from the feature classes and try to match it with a database of parts that we have stored in the system. Representative matching algorithms for the features can be found in [1]. Using the paradigm of object-oriented programming allows this method to interrogate the type of objects sent to it in its invocation. They will typically be an arbitrary aggregate of features that can be interrogated as to their type (surface, hole, cavity). The class **parts_database** contains methods that are used to extract the features from the database that the matching algorithms use. Note that the database of object descriptions is a self contained class. The algorithms work independent of the modeling primitives employed in the database. This allows for multiple representations of objects within the database.
- **orientation_part_method:** This method will try to ascertain the orientation of a part given its features. Again the features are supplied as an aggregate which then can be used in any arbitrary matching metric to estimate part orientation. The metrics themselves may be thought of as methods and the system could be subdivided into a set of complementary matching methods using different metrics.

4.2. PICK UP PART

The object class **pick_up_part,** shown in figure 4, is used to acquire the identified part. The methods are given the part label as well as its orientation vector. In the case of a dynamic scene with movable parts, the class **identify_part** could be called to re-establish the orientation parameters of the part. The methods associated with **pick_up_part** are:

- **gripper_pose_method:** This method is used to orient the manipulator and attached gripper to approach a part to be picked up. This method will need to access the class **gripper_control** to align the gripper in the desired way.
- **approach_part_method:** This method will access the class **arm_control** for the manipulator arm to approach the part that is to be picked up.
- **grip_with_force_method:** This will close the fingers of the gripper until a force threshold is reached. The threshold can be an aggregate depending upon the particular gripper used, allowing a generic method that will be implemented at the physical level differently depending upon the particular gripper configuration.
- **monitor_slip_method:** This is an asynchronous method that will be invoked and monitored until a method is called that ends it. This method will raise an asynchronous exception should a slip parameter be exceeded. This is an example of a common robotics program construct, the guarded monitor, that asynchronously senses an event. The object-oriented paradigm does not explicitly allow for this type of construct; however, it may be implemented by interfacing operating system interrupts to the program environment.

All of the **gripper_control** methods access a particular instance of a gripper, which may vary with robot arm and end effector mounted. The methods can determine which effector is available and use the appropriate lower level gripper routines associated with the physical device in use.

4.3. TRACK MOVING PART

This class, also shown in figure 4, embodies the behavior of the mobile eye. It integrates the feedback from the vision sensor with the movement of the manipulator arm. The methods it uses are:

- **calibrate_arm_method:** This method is invoked to relate the image plane to the robot's workspace.
- **find_blob:** This method will isolate a region in the image that is of interest. A variety of standard (or new) methods can be used here.
- **center_blob:** This method will attempt to keep the blob centered in the image, calculating trajectories and accessing the appropriate arm control methods.

While this is a simple integration task of two devices (camera and arm), the real-time constraints pose large problems. **Center_blob** can be implemented asynchronously to accomplish this.

4.4. LOWER LEVEL SENSOR CLASSES

The methods that are associated with the higher level tasks described above access class objects that are associated with lower level sensory tasks. The lower level sensor classes have methods that are able to deal with a variety of sensor implementations and sensor characteristics. Typically, sensor data operators are implemented in these class methods (figure 5).

4.4.1. Point Class

This class is used to establish a point contact. It has two methods depending on the sensory medium employed:

- vision_point_method: The point can be found from stereo, texture, focus, or any other vision operator capable of providing a point measure.
- tactile_point_method: This method will use the tactile sensor class to find a 3-D point contact.

The methods can be explicitly called as to which medium is to be used, a default method can be established, or a means for determining which sensor to be used can be imbedded in the methods themselves.

4.4.2. Edge Class

This class is used to identify and quantify an edge. Its methods are:

- vision_edge_method: This uses a vision sensor to establish an edge in three dimensions.
- tactile_edge_method: This will invoke the tactile sensor and associated motion algorithms of the arm to find an edge on a part.
- verify_edge_method: This meth⌐d employs both of the previous methods to use taction to ·rify that an edge determined by vision is accurate. It is a synthesis of the two previous methods.

4.4.3. Build_surface Class

Identifying a surface entails creating a model of the surface from sensory information and then entering a matching phase to find a surface on a part that is consistent with the model. Determining the surface can be done with vision alone, touch alone or by combining the two modalities. If a range finding sensor is being used, then a surface model can be created directly from the dense surface points found with the sensor. Vision methods by themselves tend to be less robust, and typically only provide sparse depth information. Touch, on the other hand, provides dense and reliable surface information. The drawback to touch sensing is the higher level of active control needed with a tactile sensor as opposed to a passive vision sensor. The class build_surface will use each of the methods described below to accomplish this task.

- stereo_surface_method: This method will use stereo imaging to match sparse depth points and interpolate a surface accordingly. The sparsity of depth points can be used as a measure on the accuracy of this surface. If the points are sparse (smooth, homogeneous surface or many ambiguous matches) then a hybrid method (described below) can be used that incorporates touch as well as vision in determining the surface properties.
- range_surface_method: This method will build a surface description from a range sensor. The data from such a sensor will typically be dense, allowing standard interpolation techniques to be used. This method will build a number of different surface models (plane, quadric, bicubic) depending upon the fit of the data to each of these surface models.
- touch_surface_method : In this method, we use the properties of the tactile sensor to determine the surface properties and build an internal model. The implementation of the method is dependent on the underlying tactile sensor used, and the method is invoked by the user in a way that is transparent to the underlying implementation of the physical sensor. If the tactile sensor is a single finger probe as was used in our previous work [2] then we can implement the method by using visual region information (invoked by this method) to establish regions that can be explored with active manipulator control and tactile feedback. This method will build a Coons' patch representation of the surface [8] which can then be used in a matching method. If the tactile sensor is a parallel jaw sensor with flat tactile sensing surfaces, then a different method is used. In this method, the surface is traced by taking tactile impressions of the surface with the pad, and from this following the principal directions on the surface to record the curvature properties in small neighborhoods of the point. This provides two sources of matching information. The first is a description of the curves that make up the principal directions and an approximation to the Gaussian map of the surface. The other is a local matching construct know as the *Dupin Indicatrix* [6]. This is a measure of the intersection of the tangent plane at a point on the surface with the surface itself as the plane is moved just above or below the surface. For an elliptic surface, in which the surface lies wholly on one side of the tangent plane at the point, the Dupin Indicatrix is an elliptical curve. For a hyperbolic point, a point whose principal curvatures differ in sign, the indicatrix is a hyperbola. If the point is an umbilic in which the principal curvatures are equal, then the indicatrix becomes a circle. Finally, for parabolic points, the indicatrix is capable of yielding a number of different curves, which can be characterized as the absence of the curves seen above.

This measure provides us with a characterization of the surface and firm matching criteria that will be provided to the matching method. One of the inherent problems with flat pad sensor technology is the inability to cope with concave surfaces. While we still can not overcome this limitation altogether, we will be able to characterize the surfaces as being of this nature while not being able to fully quantify them. The tactile sensor is well suited for establishing these measures as it has a very accurate vector sensor capable of determining the orientation of the surface at a point. This will enable quick computation of a Gaussian map of the surface.

4.5. SENSING AND ACTUATION DEVICES

The physical sensing and actuation devices are interfaced to the rest of the system by establishing them as classes with methods that are specific to each individual sensor. As different sensing devices are added to the system, the changes are made transparent to the existing code by simply adding a new sensor instance within the class, changing default attributes as needed. If a new device is very different from the existing devices, a new sensing class and possibly new methods can be added to the overall system. Figure 6 shows the overall system design for the sensors, with the physical devices connected to their appropriate methods and higher level task descriptions.

5. IMPLEMENTATION

This framework is currently being implemented in the AML/X programming language [10]. AML/X is an experimental language that has been designed with manufacturing and design applications in mind. It currently contains support of object-oriented programming, including classes, methods and operator overloading. It does not however have a defined inheritance hierarchy or concurrency as integral parts of the language. Inheritance schemes vary greatly among object-oriented languages. The majority of these schemes rely on a strict tree-like inheritance scheme or allowing the programmer to explicitly specify the relationships between classes. We have observed that robotic tasks involving any complexity exhibit network as opposed to tree-like connections. Asynchronous operations can be implemented using the operating system interface but are also not a language primitive. As the system described here evolves, AML/X may be extended in ways that will support a richer inheritance scheme and possibly concurrency.

6. SUMMARY

This paper has described a programming framework for implementing multi-sensor robotic tasks. The framework shows promise in being able to link actuation with intelligent sensor based control. Of particular importance is the ability to explore new methods involving sensors and sensing strategies in a modular and understandable way. Our current plan is to extend the framework to include multiple arm control and coordinated movement for robotic assembly.

References

1. Allen, Peter, "Object recognition using vision and touch," Ph.D. Dissertation, University of Pennsylvania, September 1985.
2. Allen, Peter, "Sensing and describing 3-D structure," *Proc. IEEE Conference on Robotics and Automation*, pp. 126-131, San Francisco, April 7-10, 1986.
3. Brady, Michael and Richard Paul, *First International Symposium on Robotics Research*, M.I.T. Press, 1984.
4. Brooks, Rodney A., "A layered intelligent control system for a mobile robot," *3rd ISRR*, Gouvieaux, France, October 1985.
5. Chiu, Stephen L., David J. Morley, and Jim F. Martin, "Sensor data fusion on parallel processor," *Proc. IEEE conference on Robotics and Automation*, pp. 1629-1637, San Francisco, April 7-10, 1986.
6. DoCarmo, Manfredo, *Diffential geometry of curves and surfaces*, Prentice-Hall, Englewood Cliffs, NJ, 1976.
7. Erman, Lee, Frederick Hayes-Roth, Victor Lesser, and D. Raj Reddy, "The HEARSAY-II speech-understanding system: Integrating knowledge to resolve uncertainty," *Computing Surveys*, vol. 12, no. 2, pp. 213-253, June 1980.
8. Faux, I. D. and M. J. Pratt, *Computational geometry for design and manufacture*, John Wiley, New York, 1979.
9. Henderson, Thomas C. and Chuck Hansen, "Multisensor Knowledge Systems," Technical Report, University of Utah, 1986.
10. Nackman, Lee, Mark Lavin, Russell Taylor, Walter Dietrich, and David Grossman, *AML/X: A programming language for design and manufacture*, IBM T.J. Watson research center report RC 11992, July 1986.
11. Shafer, Steven A., Anthony Stentz, and Charles E. Thorpe, "An architecture for sensor fusion in a mobile robot," *Proc. IEEE conference on Robotics and Automation*, pp. 2002-2011, San Francisco, April 7-10, 1986.
12. Stefik, Mark and Daniel Bobrow, "Object-oriented programming: Themes and variations," *AI Magazine*, vol. 6, no. 4, pp. 40-62, Winter 1986.
13. Taylor, Russell H., James U. Korein, Georg E. Maier, and Lawrence F. Durfee, "A configurable system for automation programming and control," *Proc. IEEE conference on Robotics and Automation*, pp. 1871-1877, San Francisco, April 7-10, 1986.

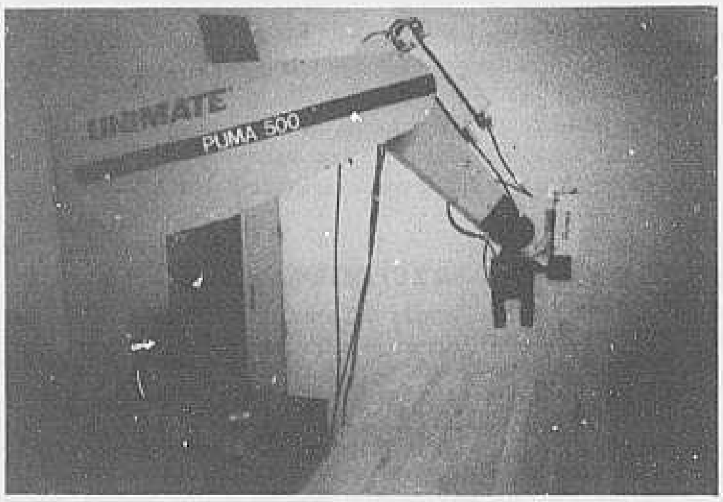


Figure 1: Experimental system.

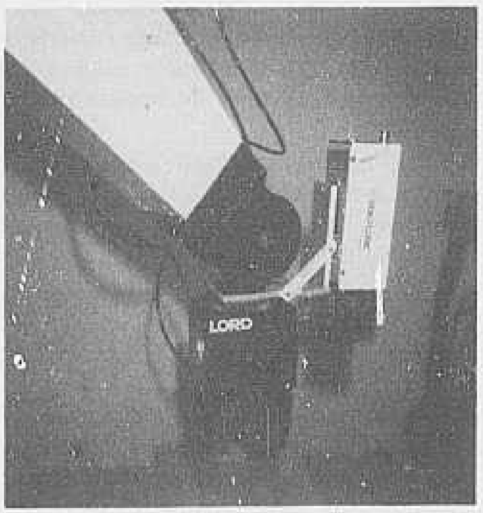


Figure 2: Parallel jaw gripper/tactile sensor.

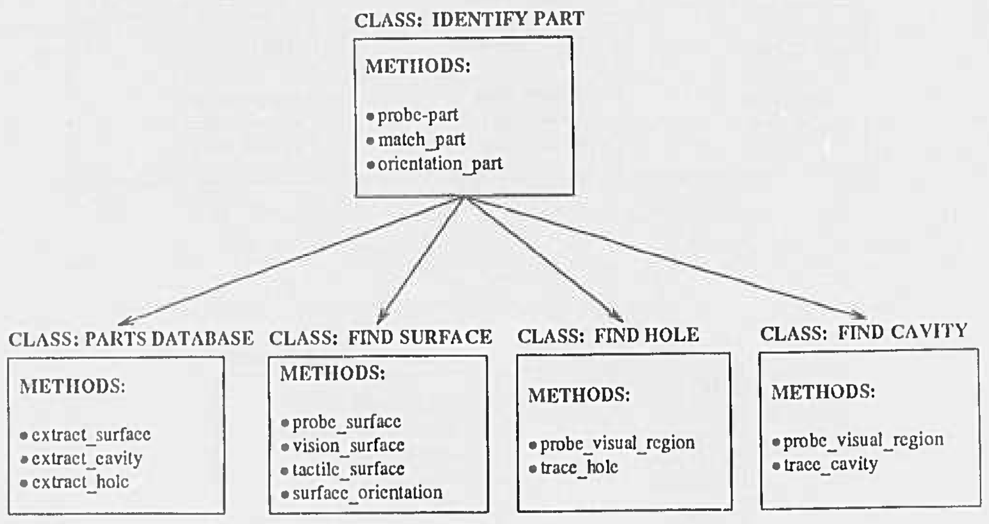


Figure 3: Identify part task.

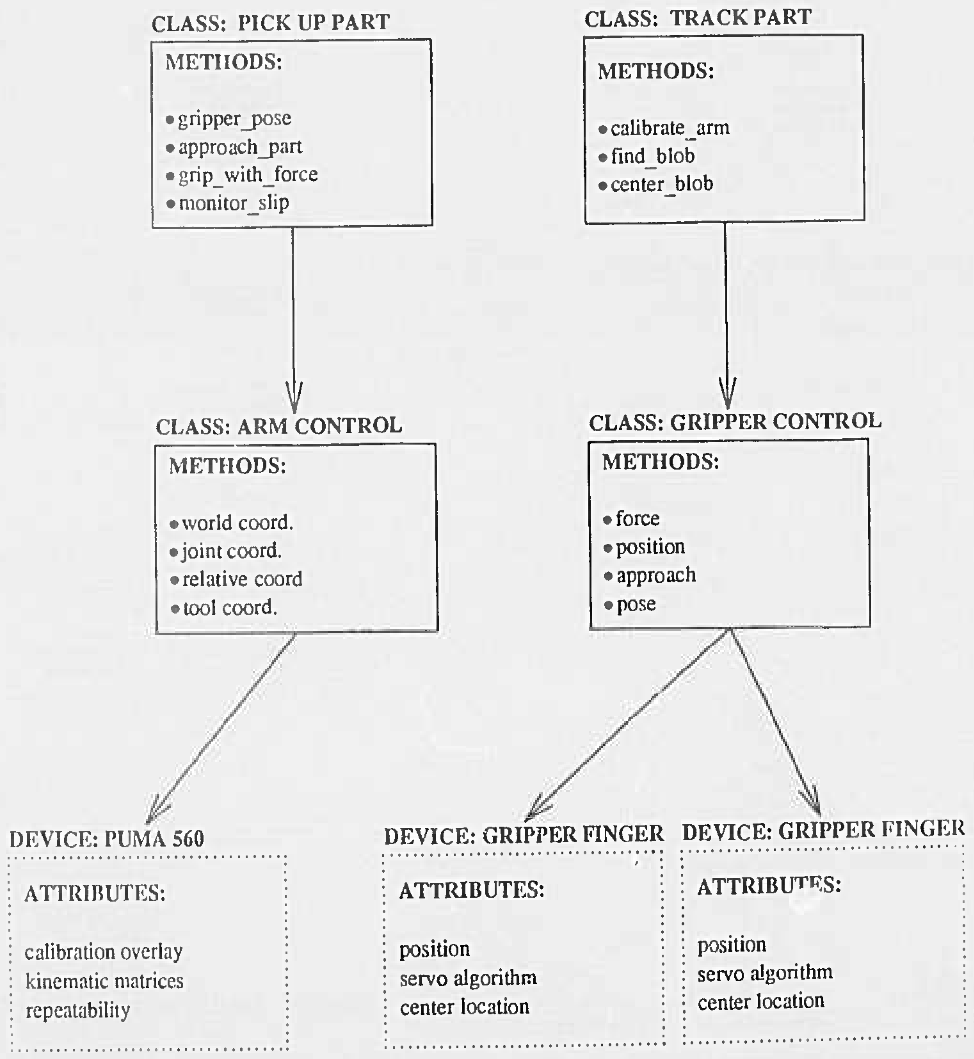


Figure 4: Pick up part and track part tasks.

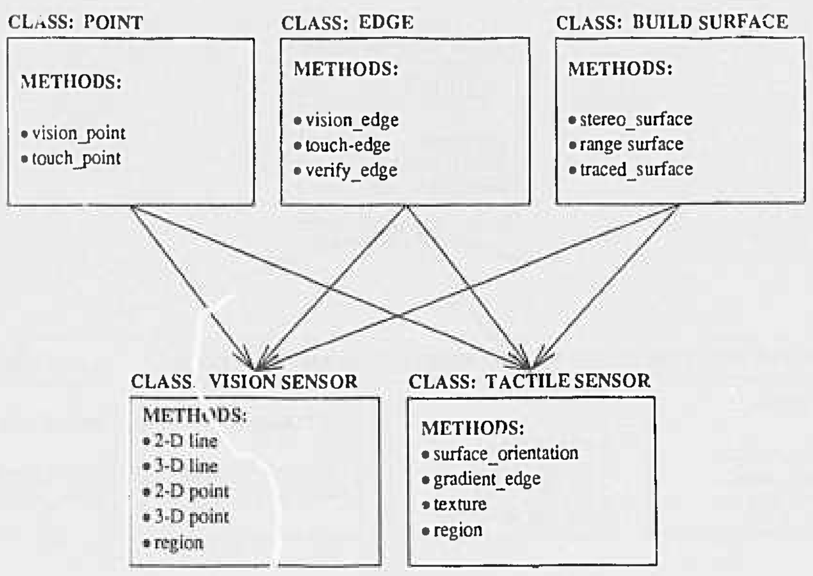


Figure 5: Low level sensor classes.

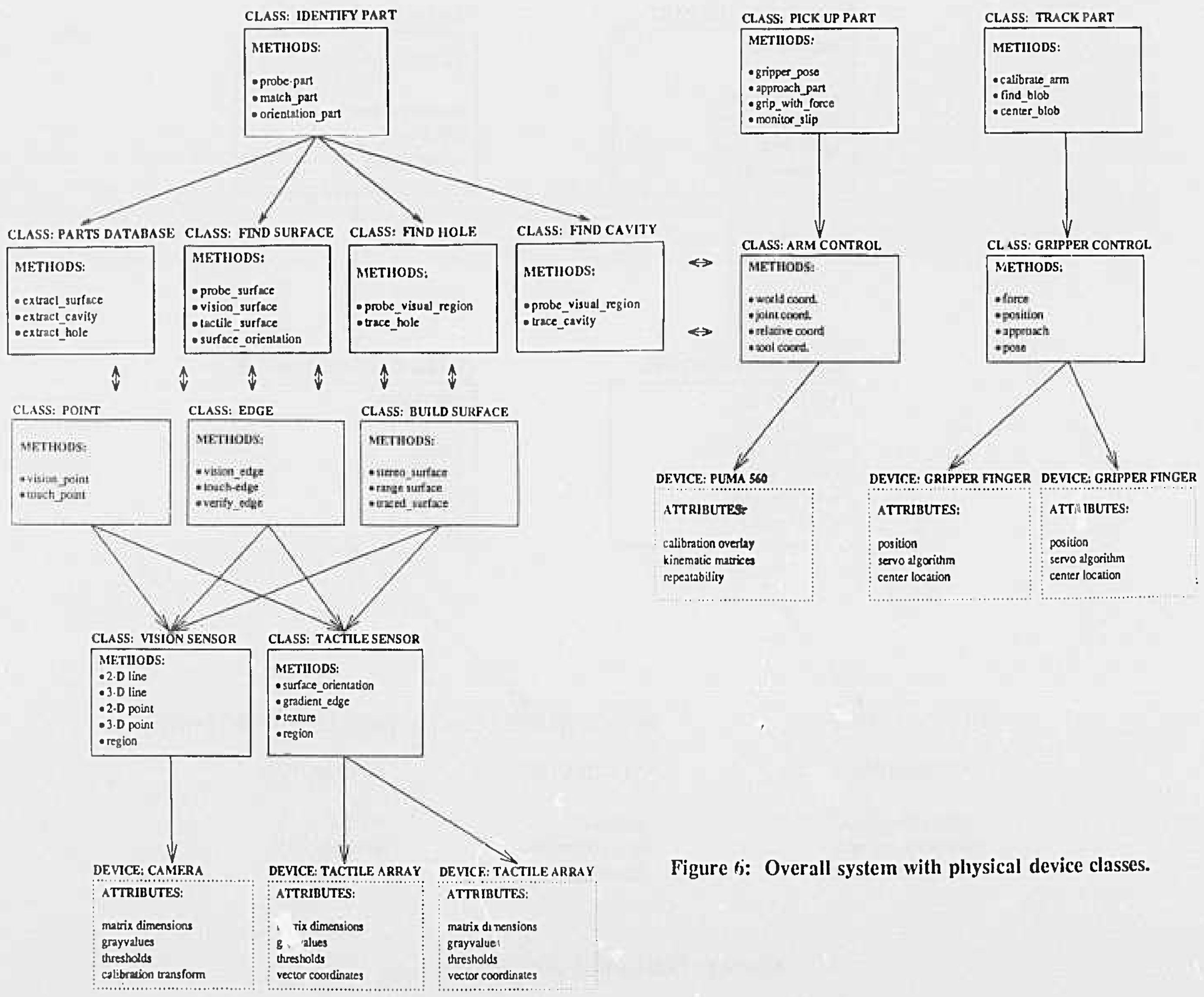


Figure 6: Overall system with physical device classes.

SHAPE FROM DARKNESS: Deriving Surface Information from Dynamic Shadows

John R. Kender[1]
Earl M. Smith

Department of Computer Science
Columbia University
New York, New York 10027

ABSTRACT

We present a new method, *shape from darkness*, for extracting surface shape information based on object self-shadowing under moving light sources. It is motivated by the problem of human perception of fractal textures under perspective. To introduce the method, one-dimensional dynamic shadows are analyzed in the continuous case, and their behavior is categorized into three exhaustive shadow classes. The continuous problem is shown to be solved by the integration of ordinary differential equations, using information captured in a new image representation called the *suntrace*. The discretization of the one-dimensional problem introduces uncertainty in the discrete *suntrace*; however it is successfully recast as the satisfaction of *8n* constraint equations in *2n* unknowns. The extension to two dimensions is straightforward: there are *16n* equations in *2n* unknowns. A form of relaxation appears to quickly converge these constraints to accurate surface reconstructions; we give several examples on simulated images, both one- and two-dimensional. The shape from darkness method has two advantages: it does not require a reflectance map, and it works on non-smooth surfaces. We conclude with a discussion on the method's accuracy, its relation to human perception, and its future extensions.

1 Introduction

We present a new, active method for obtaining shape information from low level cues. It exploits the information implicit in the shadows that an object or an object part casts upon itself or another object. In spirit, it is most like the photometric stereo method of Woodham (Woodham, 1981), in that it requires knowledge of the illuminant position. However, it also extends the existing work on shadow geometry of Shafer (Shafer, 1985) and others, and gives additional insight into the nature of shadows, especially in the cases where the objects are neither polyhedra nor smooth, or where the shadows are dynamically changing. The method has two major advantages. It appears to work best for textured objects, that is, where existing methods fail most badly. And it is more robust than existing methods, in that it requires little a priori information about a surface's reflectance. Further, it illustrates the inherent utility--and complexity--of static or dynamic shadow-based cues for any integrated vision system, whether active or passive.

2 Historical Background

The method, which can be called *shape from darkness*, was motivated by an interest in the human perception of fractal textures. As Pentland has shown (Pentland, 1984), the fractal dimension of textured surfaces is a powerful feature on which the segmentation of an image can be based. He further observed that the image of a single fractal surface viewed under perspective has non-constant fractal dimension. It is conjectured that this change in measured feature is closely related to the change in overall local surface orientation of the surface with respect to the observer. If this is the case, then fractal dimension can serve as a basis for a "shape from fractal" method, similar to other gradient-based shape from x methods.

[1]This research was supported in part by ARPA grant #N00039-84-C-0165, by a NSF Presidential Young Investigator Award, and by Faculty Development Awards from AT&T, Ford Motor Co., and Digital Equipment Corporation.

However, the mathematics behind such relationships appear formidable. This is because the observed change in fractal dimension appears to be due to the increasing self-occlusion of the fractal surface as it is viewed at increasingly oblique angles. That is, unlike an airborne observer of a mountain range, an observer down in the foothills sees very little of the mountain peaks: he sees mostly the sides of foothills. The mathematical difficulty stems from the intractability of the threshold-like non-linear functions that express the nature of object occlusion; the difficulties are similar to the ones faced when trying to integrate object segmentation with standard shape-from-x methods.

Nevertheless, the problem does have the following analogue, which ultimately suggested the method reported here. It is that self-occlusion is very similar to shadowing: were a light source moved to the observer's position, the self-occluded areas would now be the ones in shadow. Thus, instead of attempting to investigate the effect that varying surface orientations have on observed fractal properties (or, equivalently, the effect that varying observer positions have), one can explore the effects that varying light source positions have on the generation of a fractal's shadows. Ideally, one would like to look into the shadows in order to see what information has been lost.

Generating and analyzing shadow information allows for several computational efficiencies. Essentially, when working with shadows, one is doing rendering and shading under extreme conditions. The capture of shadow information from real imagery or the generation of shadows synthetically both result in binary imagery. Instead of collecting shading information that has a range of values, one obtains a characteristic function instead: zero means shadow, one means illuminated. Simple thresholding of actual imagery is usually all that is required, and the synthetic casting of shadows is a straightforward computation. The imagery that results can be seen as extreme shape from shading in another sense. A synthetic shadow image can be obtained in the standard graphic rendering way by first thresholding the reflectance map: all gradients which reflect any light at all are set to one, and the remainder of the map stays at zero (for self-occluding). What results when an image is rendered with such a map is an image with extreme contrast; indeed, the contrast cannot be more extreme.

Recovering the depth or orientation of those surface fragments that have been shadowed is clearly a difficult task given only one shadow image. As with many other problems in vision, many influences are conflated into the simple image observable, the shadow. The beginning of a shadow is determined not only by surface orientation and illuminant direction, but also by the absence of any prior surface to overshadow it. The termination of a shadow depends on the relative heights and orientations of both the shadowing and shadowed surface. Deconflating these influences in a single image is not necessarily impossible; it depends on the additional information and assumptions one also brings to the task. For example, if it is known that the surface is that of a hemisphere, its position and radius are easily recovered, even without knowledge of the illuminant direction. Less restrictive assumptions, such as the surface having a band-limited fourier spectrum (and therefore "smooth" in exactly this sense of smooth), may also admit to solutions, perhaps in a form analogous to the Logan theorem characterizing a signal by its zero-crossings (Logan, 1977). But still weaker assumptions, such as the surface simply being twice differentiable, probably do not lead to solutions at all. This is because smoothness as defined by differentiability is the assumption implicit in true shape from shading, and true shape from shading depends heavily on the amount of curvature in the reflectance map (Lee, 1985); the thresholded reflectance map has none.

3 Problem Formalization

The shape from darkness problem is more straightforward to solve by using multiple images. The observer and the objects can be held stationary, obviating any image-to-image correspondence problem, and what is moved is the light source, in a manner similar to photometric stereo. Photometric stereo usually can be done with three illuminant positions, although four is the usual number used in practice in order to prevent exactly the problem discussed here: objects in self-shadow. It is apparent that even four shadow images is woefully inadequate for shape from darkness under reasonable surface assumptions. Thus, the problem is relaxed to allow a fixed number of illuminant positions, the exact count and location of which are to be determined. The added complexity of increased imagery is mitigated in part by its binary nature, and in part by the lack of any necessity to calibrate the shadow reflectance map, since the latter is determined solely by the illuminant direction. One only needs to define the location of the shadow terminator orientations.

For simplicity in the discussion that follows, the problem is further reduced to its natural one-dimensional subproblem. That is, the algorithms presented here will discuss the recovery of a planar curve rather than a surface, given illuminants that lie in the plane of the curve. (The extension of the method to the full two dimensional case, including a discussion of the degrees of freedom of illuminant placement, is sketched later.) Thus, we assume that depth is a function solely of x, z=f(x), rather than z=f(x,y), and that the illuminants lie wholly within the xz plane. Note that photometric stereo has a similar one-dimensional analogue, with one-dimensional reflectance maps that are functions of curve derivative rather than of surface gradient. In one-dimensional photometric stereo, three lights are necessary to prevent objects--here, curves--from self-shadowing.

4 The Continuous Problem

It is instructive to consider the shape from darkness problem as a continuous problem first. Assume that the illuminant is an infinitely distant point source, and that the observer is infinitely far in the positive z direction. (Thus, instead of investigating the surface properties of a fractal seen under perspective, we are now exploring the recovery of curve information from shadows generated under parallel illumination.) Given that the illuminant will appear in many orientations, it will be convenient to identify the illuminant with the sun, the positive direction of the x axis with the east, illumination at zero slope with dawn, illumination at positive slopes with morning, and illumination from the positive z axis with noon; often these terms are more immediate and compact.

As shown in the figure, it is easy to show that under these conditions all curve points fall into one of three classes of dynamic shadow behavior under increasing morning illumination, with analogous classes in the afternoon. A point either can become illuminated because it gradually is moved out from self-shadowing, or it can be always illuminated, or it can become illuminated because it gradually moves out from a cast shadow. These definitions can be made precise, at the given points:

A *minus* point m has f'(m) >=0 such that for all x, x>m implies f(x) <= f(m) + f'(m)(x-m). (Implicitly, f''(m) <= 0.) Intuitively, a minus point can only be in shadow when it is (or would be) self-shadowed. It becomes illuminated precisely at the time of day when the rising illuminant's slope is equal to f'(m). When f(m) becomes illuminated, points to the immediate west of m remain in shadow; therefore, in the direction of illumination the transition at m is from illumination into darkness. This terminator travels west with increasing illumination, and crosses descending values of f. Note that the shadow is caused by light grazing f(m), and it is therefore diffuse, especially at low illuminant slopes. Such a point is therefore called minus for five negatively flavored reasons (a sixth becomes apparent shortly): its second derivative is negative, its terminator goes from light to dark, the terminator travels west, the terminator descends, and the shadow is not sharp.

A *zero* point z is such that for all x, x>z implies f(x) <= f(z). (Implicitly, f'(z) <= 0.) Intuitively, a zero point is never shadowed (in the morning), not even at dawn. It becomes illuminated when the rising illuminant has slope equal to zero. It never experiences a terminator: it is characterized by zero shadow and zero change.

A *plus* point p is every other point. Negating and manipulating quantifies yields: either f'(p) <= 0 and it is not a zero point, or f'(p) >= 0 and it is not a minus point. Intuitively, a plus point can only be shadowed due to cast shadows. It becomes illuminated when the rising illuminant grazes a minus point at m (illuminant slope is f'(m)), such that f(m) = f(p) + f'(m)(m-p). When f(p) becomes illuminated, points to the immediate east of p remain in shadow; therefore, in the direction of illumination the transition at p is from darkness into illumination (thus, plus). This terminator travels east (plus) with increasing illumination. Note that the shadow is caused by occlusion, and is therefore sharp (plus). (However, f''(p) is not necessarily positive, and the terminator does not necessarily cross ascending values of f.)

The function f can therefore be partitioned into segments and the segments labeled by their shadow class. The grammar of segment labels is simple; in the morning it is given by the regular expression $((+-)^*0)^*$. Such strings have three significant transitions. Plus to minus occurs at f'' = 0 with f' at a local maximum. Minus to zero occurs at f' = 0 with f at a local maximum. Minus to plus occurs at curious "second grazing" points, those points m where f'(m) is equal to the illuminant slope, but where there is also a p>m with f'(p) also equal to the illuminant slope, and f(p) = f(m) + f'(m)(p-m). (The fourth transition, zero to plus, appears to have no special significance.)

4.1 The Continuous Suntrace

Quantitative reconstruction can be based on the integration of the derivative information intrinsic in the minus points. The reconstruction requires an additional representation of image information, called the *suntrace*, from which the requisite derivative information is obtained.

The suntrace is a mapping from the domain of the original curve into (morning) illumination slopes. For each x, it records the slope at which the value f(x) first became illuminated. The suntrace is a function of x, since a given f(x) can become illuminated only once. Depending on the underlying curve, the suntrace may be unbounded: although the entire curve must be illuminated no later than noon, noon corresponds to an unbounded illumination slope.

Since zero points are illuminated at dawn, they have suntrace values identically zero; see Figure 1. Minus points are likewise easy to detect and label: they are exactly those points (in the morning) with negative (minus) suntrace derivatives, since their terminators move west with increasing illuminant slope. What remains are the plus points; they have positive (plus) suntrace derivatives.

4.2 Solution Using ODEs

Given a morning suntrace, the underlying curve can be partially reconstructed. A contiguous curve segment with minus labels can be integrated into a function segment by using the suntrace value of the point as the value of f' at the point. The segment, however, must "float" at an unknown height until it is given an absolute height by the appropriate constant of integration.

By definition, the function values of all plus points can be determined relative to the position of their corresponding minus points that shadow them. For a plus point p, the calculation is based on the relation f(m) = f(p) + f'(m)(m-p), where the corresponding minus point m is found in the suntrace as the least m greater than p that has the same illumination slope, f'(m), that p has. Entire contiguous segments of plus edges can therefore be fixed in space, and joined to their integrated minus segment

The now completed plus-minus complexes can themselves be joined one to another at their common "second grazing points" (that is, at minus-plus transitions). In this way, long, self-consistent segments of the curve result, but with each "floating" with respect to a constant of integration; see Figure 2.

The fuller recovery can never be made since a simple morning suntrace provides no information about zero points. Their relative and actual depths can attain arbitrarily high values, and any self-consistent segments separated by zero points can freely float relative to each other, as long as the slope of the intervening zero segments remain negative.

Pinning down the constant of integration and restricting the behavior of zero points can be achieved by using a second suntrace, usually the afternoon suntrace which maps illumination slopes from noon to dusk. It is apparent that the only point that can be labeled a zero point for both suntraces is the global maximum. All other points are shadowed at least once and can therefore be assigned a function value relative to some constant. What results, within the accuracy of the suntrace and the integration, is a reconstruction of the underlying curve with depth values relative to a single constant of integration: the global maximum.

5 The Discrete Problem

Discretizing the shape from darkness problem requires some care. The heart of the difficulty is twofold. A discrete suntrace, however fine, can only give upper and lower bounds to function derivatives for minus points. Further, given digitization, it is not always clear where the shadowing function values in the discrete suntrace really ought to be. Indeed, as the method below describes, occasionally two different function values will serve to set the bounds on the derivatives, one for the upper bound and one for the lower bound, since the true minus point may be somewhere between them.

Rather than attempt to approximate a derivative for each minus edge, the following method attempts to maintain solution accuracy by calculating the exact upper and lower bounds the solution could have, given the discretization of the suntrace that it starts with.

5.1 The Discrete Suntrace

Any given function value f(x) will become illuminated only once. Thus, there will be a time of morning, t, at and before which f(x) is in shadow, and after which, at t+1, it becomes illuminated. Call the function value that shadowed f(x) at time t (but could not shadow it at time t+1) the *last shadower*. The last shadower may not be the only function value that fails at time t+1 to shadow f(x); call the most prominent shadower the *failing shadower*. It is not hard to see that for illumination in the morning, the west-to-east order of the function values is f(x), failing shadower, and last shadower, although there may be many other undistinguished function values scattered amongst these three. These two shadowers generate important constraints on the upper and lower bounds of f(x); more interestingly, f(x) itself feeds back constraints on the upper and lower bounds of its shadowers, too. It is important to note that these two shadowers of f(x) may be the last and failing shadowers of other function values, too; their roles at these other points may even be reversed. See Figure 3.

The shape from darkness method begins by collecting from the discrete suntrace, for every element x in the domain of the curve, information about such shadowers. The last shadower of f(x) is found in the morning in the following way. If f(x) first became illuminated at time t+1, the last shadower of f(x) was the nearest eastern illuminated neighbor to f(x) at time t. The failing shadower of f(x) is the nearest eastern illuminated neighbor at time t+1.

Fortunately, such information can be collected in one pass through the suntrace. Assuming both a morning and afternoon suntrace, each element x of the domain will gather eight pieces of information: for each of the four morning or afternoon last or failing shadowers, it stores their position and the time of their shadowing (t or t+1).

5.2 The Eight Constraints per Point

Given this information, each point in the domain affects and is affected by these four critical shadowers. Each point therefore participates in eight constraints, four to do the affecting, and four to be affected by. Given that the morning and afternoon suntraces are completely symmetrical, there are only four basic conceptual relations: forward or backward constraints on upper or lower bounds. The forward constraints propagate constraint information in the direction of the illuminant; the backward constraints propagate it against the illuminant.

The forward constraints are based on the following observations. At point x, x's upper bound can be no higher than the projected shadow of the upper bound of its last shadower. (If x's upper bound were any higher, x would not be shadowed at time t). Similarly, at point x, x's lower bound can be no lower than the projected shadow of the lower bound of its failing shadower. (If x's lower bound were any lower, x would instead be shadowed at time t+1).

In the morning, the forward constraint equations are therefore:

UPPER $u(x) <= u(ls(x)) - (ls(x)-x)*sls(x)$

LOWER $l(x) >= l(fs(x)) - (fs(x)-x)*sfs(x)$

where u(.) and l(.) represent the upper and lower limits in effect at any time, ls(.) and fs(.) are the coordinates of the last shadower and failing shadower, and sls(.) and sfs(.) are the illumination slopes at the times of last shadow and failing shadow.

The backward constraints are a bit trickier, but it is their feedback that seems to account for the method's power. Consider the upper bound at x. Since the failing shadower must fail to shadow x, the upper bound of the failing shadower is limited by the height at which it just barely fails to shadow x; the maximum allowable height for the failing shadower occurs when x itself is at its maximum. (If the failing shadower's upper bound were higher, it would instead shadow x.) This height can be determined by backprojecting the upper bound of x along the slope in effect at the failing shadow time, t+1. Similarly, consider the lower bound at x. Since the last shadower must successfully shadow x, the lower bound of the last shadower is limited by the depth at which it just barely succeeds in shadowing x; the minimum allowable depth for the last shadower occurs when x itself is at its minimum. (If the last shadower's lower bound were smaller, it would instead fail to shadow x.) This height can be determined by backprojecting the lower bound of x along the slope in effect at the last shadow time, t. See Figure 4.

In the morning, the backward constraint equations are therefore:

UPPER $u(fs(x)) <= u(x) + (fs(x)-x)*sfs(x)$

LOWER $l(ls(x)) >= l(x) + (ls(x)-x)*sls(x)$

Four similar constraints apply to the information gathered for x from the afternoon suntrace.

It is surprising that these appear to be all the constraints possible (aside from the trivial constraint that $u(x) > l(x)$). Other relationships between the upper and lower bounds of x, upper and lower bounds of its last shadower, and upper and lower bounds of its failing shadower, do not appear to be constraining. For example, if x's upper bound decreases, it has no effect on the upper bound of its last shadower.

5.3 Solution Using Relaxation

The specific family of constraints that result from a given suntrace have a complex interrelated structure. It is not apparent whether there is any special solution method applicable to this problem in general, or even for well-defined subclasses of curves. There are $8n$ inequalities in $2n$ unknowns, and there is a well-defined objective function to minimize: that is, the sum, over all x, of u(x) - l(x)).

Although the problem might be solved using linear programming, a more attractive solution method is the use of a version of relaxation. Conceptually this consists of a number of successive iterations, in each of which the eight constraint equations are successively applied to each point x in the domain. If the application of any constraints results in better estimates for u(x) or l(x), they are updated. As in the continuous case, the only valid initial values are those of the global maximum (the only point labelled zero in both suntraces); its upper and lower limits are set arbitrarily to a pleasant value (say, zero) before the relaxation begins.

In practice, convergence seems very rapid. Unlike some relaxation algorithms, updating is based on thresholds, so upper and lower bounds are only altered if they are moved closer together. The method is therefore more likely to terminate when it recognizes a lack of measurable progress.

Extension of the algorithm to two-dimensional surfaces is straightforward. A two-dimensional suntrace is still binary. The constraint equations easily decompose into two families of x-based ("east-west") and y-based ("north-south") constraints. In all, each point is affected by 16 constraints, and convergence appears even more rapid than in the one-dimensional case. There is a great deal of freedom in selecting how the illuminant can be positioned; for simplicity, two orthogonal passes suffice.

6 Experimental Results: One-Dimensional

In the experiments that follow, some of the generalities of the algorithm were made particular. For ease of comparing the final reconstructed curve to the original, the global maximum of the reconstruction was initialized to its true known height. Sun positions were simulated at constant slope increment; thus, sun angles in the morning linearly increase in tangent. (Under this scenario, the sun literally rises, rather than travels an arc!) This policy of constant increment seems to be closely related to the encouraging accuracy obtained in the final processing step, where the final estimate of the curve is defined to be the curve midway between the computed upper and lower bounds.

Each of these series of test images shows the following.

The first figure of a series is the original curve, with its morning and evening suntraces. The domain of the original curve is aligned with the domain of the suntraces. Both suntraces have the axes for increasing sun slope pointing toward the curve. Thus, on all suntraces, the line nearest the curve is pure black, indicating all pixels have been illuminated.

The second figure of a series is a record of the constraint processing. Initial estimates for upper and lower bounds as propagated from the global maximum have gradually approached each other, subject to the suntrace data.

The third figure of the series shows final upper and lower bounds, the original curve, and the superimposed best estimate.

The first series is an image of a self-similar mountain. It is approximately 300 points wide by 85 points peak-to-peak. The suntrace was taken at increments of .1, that is, at approximately four degrees, to a maximum of 30 increments. The final estimate has a cumulative total error of less than 68 (about .2 error per pixel, average), and a maximum single point error of less than 1.2.

The second series is the same image, but with a suntrace increment of 1: that is, the first non-dawn suntrace is taken at 45 degrees, and only four increments are possible. Although not a realistic test, it demonstrates more visibly the method and its results, especially the goodness of the final estimate even under extremely severe conditions.

The third series demonstrates the applicability of the processing to very smooth imagery: a semicircle of radius 50, again under 30 increments of .1 each. Maximum error occurs at the extreme left and right of the "table", although reconstruction error within the circle is no more than 0.5.

7 Experimental Results: Two-Dimensional

A fourth and fifth series of experiments show the reconstruction of a two-dimensional fractal surface, and of a random surface, respectively. As in the prior examples, the reconstructed surface was normalized by properly offsetting its starting global maximum in order to ease the comparison of the result with the original. Note that unlike the one-dimensional case, it is possible for many two-dimensional points to be forever unshadowed: the choice of any of these as the starting value will do.

In each of the two following series, the first figure is an intensity-encoded depth map of the surface. The second figure is a similarly encoded reconstruction. (Suntraces have been omitted.) Reconstruction was so highly accurate that the third image is the gross error multiplied by a factor of *ten*.

The fourth series shows a fractal surface created by the tensor product of the first series curve with itself, as is evident from its symmetry. (It is not a true fractal, in that the x and y dimensions are highly correlated, but it certainly is not a smooth surface.) The tensor product is remapped to a range of 25 to 225. Suntrace data was taken with an increment of .3. Reconstruction took 16 iterations, with average error of .16.

The fifth series shows a random surface, with pixel depths uniformly distributed in the range 27 to 227. Suntraces were taken with increments of 1.0, and convergence was attained in 14 iterations, with average error of .96.

8 Discussion

8.1 Performance

It appears that the accuracy of the final estimate is surprisingly good, and may be related to the use of constant illumination slope increment. Choosing the midway curve is guaranteed to minimize worst case error, since the midpoint can never be off more than half the available range.

Aside from the empirical data given above, little is known about the theoretic performance of the algorithms except in two worst cases. In terms of accuracy, the worst case image occurs in a monotonically decreasing function with positive curvature (as in $z = 1/(x+c)$). Here, points at the extreme asymptotic end have little opportunity for feedback, so the range between upper and lower bounds is virtually the same as the initial forward constraints, length*(slope(t+1)-slope(t)); if slopes increase in constant increments, this is simply length*increment. In terms of convergence, it appears that certain square wave trains takes n iterations, where n is the number of pulses in the train.

Shape from darkness has several advantages, most notably that it can exploit the surface information implicit in a class of dynamic shadows, with very little restrictions placed on the class of surfaces being shadowed: they need not be smooth. In particular, it can probably be useful in increasing the accuracy with which finely textured surfaces are viewed, especially under oblique illumination. It can also exploit smart cameras that run-length encode the incoming binary shadow imagery, but the exact information content of a shadow image, especially with respect to the information content in a gray scale image, remains to be explored.

The complexity of the data interaction does suggest why humans do not appear to derive much surface information from dynamic shadows. The necessity to store, in effect, an entire suntrace is probably excessive. On the other hand, if our earth rotated much faster (say, once every three seconds), there may have been more reason for natural systems to develop at least an approximate solution to the shape from darkness problem.

8.2 Extensions

The method admits of many extensions. The application of the method to real imagery must address the difficulties of specularity, mutual illumination, and diffuse shadows. However, in a robot environment, much of the environment can be structured to make the problem easier. For example, having the knowledge that the object is on a fixed table at a given depth can aid in the setting of lower bounds. Experiments are in progress that exploit just such constraints. In practice, what appears most critical is accuracy in illuminant position and smoothness of illuminant intensity throughout its transits. Thus, rather than the sun "rising", it is better to have it sweep through a circle: otherwise the inverse square law makes shadow-detection unnecessarily adaptive and complex.

The time required for processing two-dimensional surfaces is probably the most critical area of investigation. The problem can probably be decomposed for parallel processing in ways beyond the trivial one of partitioning the images in strips parallel to the illuminant direction; it may even be done in a hierarchical way. Selecting optimal sun positions with two degrees of freedom is challenging. Although two simple perpendicular transits allow the problem to be elegantly decomposed into two one-dimensional ones, other illuminant positions may be preferable. For example, shadow data of the surface of the earth taken periodically during a summer and a winter day have sufficient north-south sun variation to supply the second dimensional constraint. However, problems with illumination are especially acute if sun and observers are allowed to be near, and observers are allowed to view off the normal axis; this is once again the original problem of fractals under perspective.

9 Acknowledgements

Paul Douglas provided an early version of the code for the generation of suntrace information, and coined the name of the representation.

References

Lee, D. *A Provably Convergent Algorithm for Shape from Shading*, pages 489-496. , December, 1985.

Logan, B.F. Information in the Zero-Crossings of Bandpass Signals. *Bell System Technical Journal*, 1977, *56*, , 487-510.

Pentland, A.P. Fractal-Based Description of Natural Scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, November 1984, *PAMI-6*, *(6)*, 661-674.

Shafer, S.A. *Shadows and Silhouettes in Computer Vision.* Hingham, MA: Kluwer Academic Publishers, 1985.

Woodham, R.J. Analysing Images of Curved Surfaces. *Artificial Intelligence*, August 1981, *17*, *(1-3)*, 117-140.

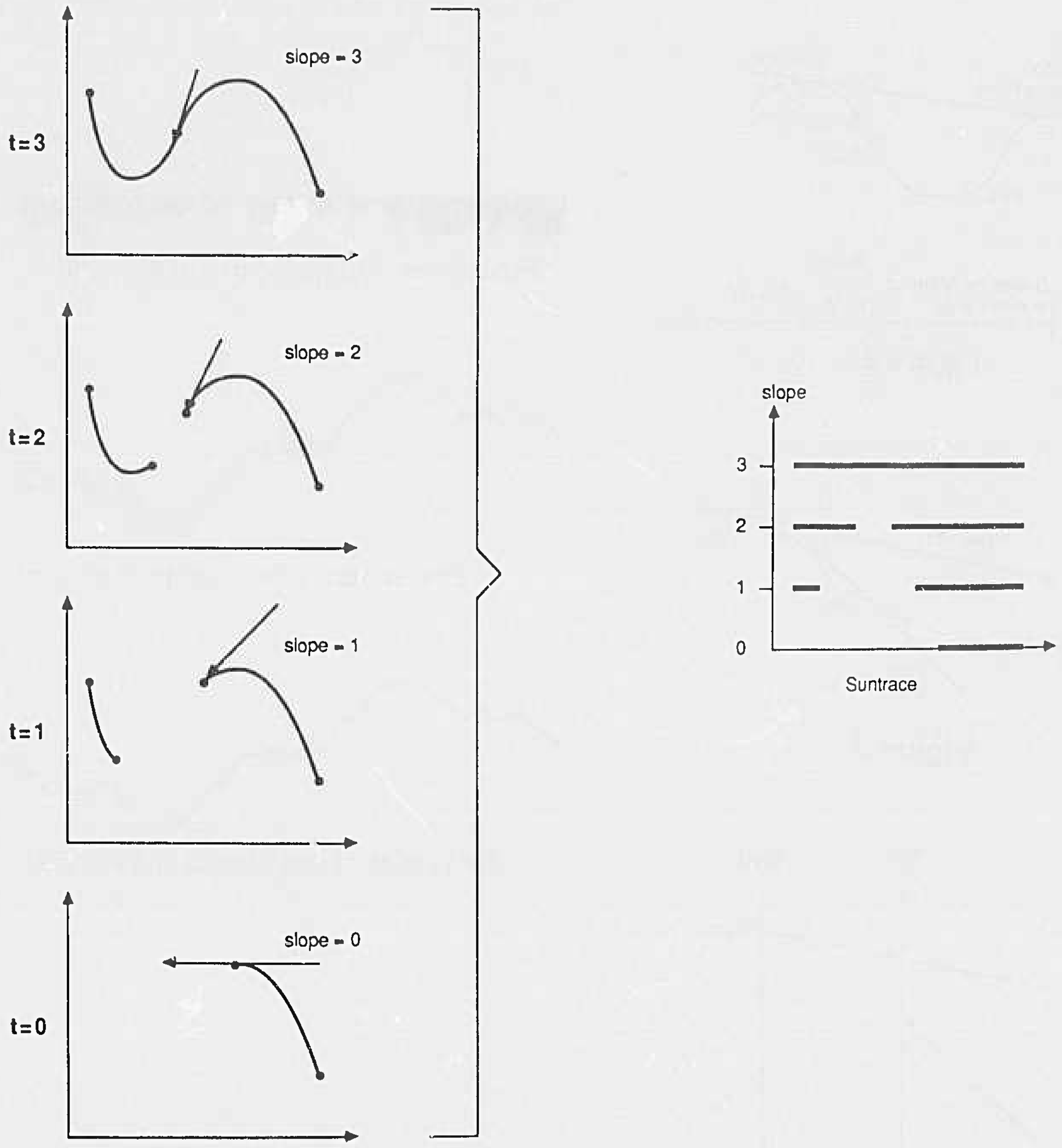


Figure 1

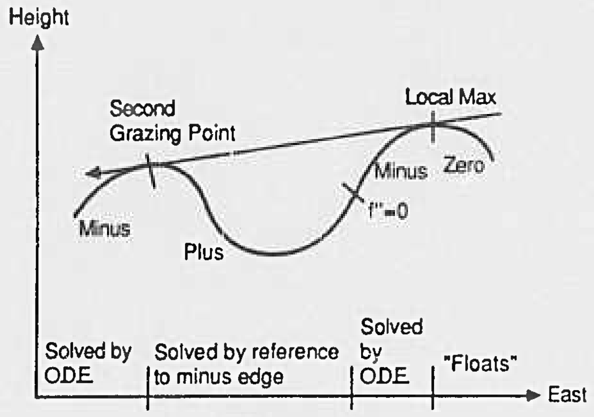
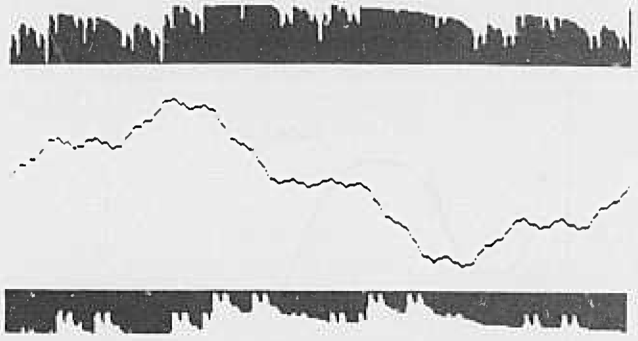


Figure 2



First series: Original curve and suntrace

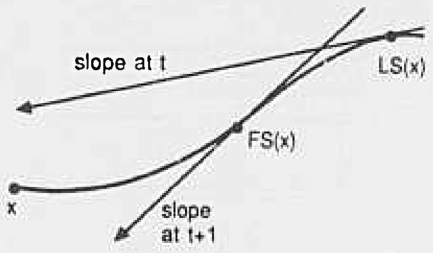
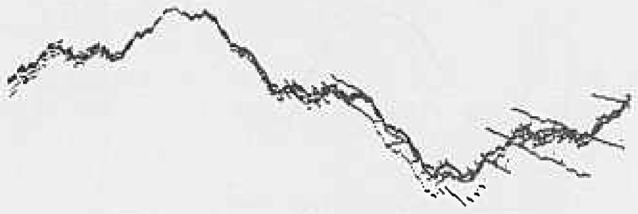
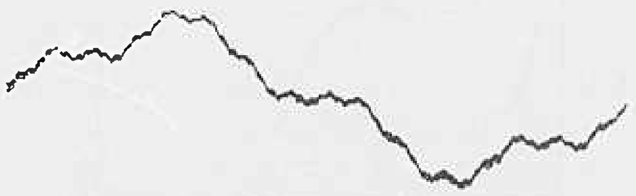


Figure 3



First series: Constraint propagation



First series: Final bounds and estimate

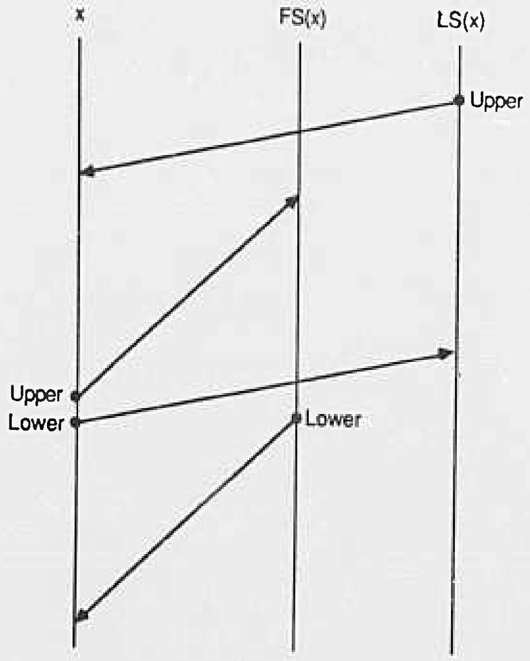
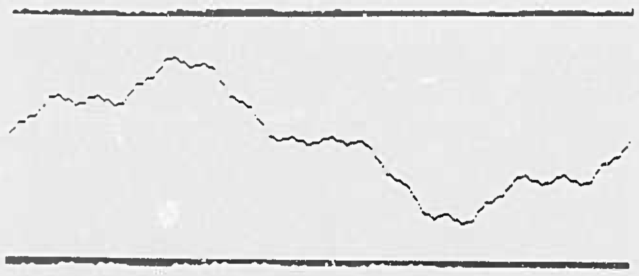
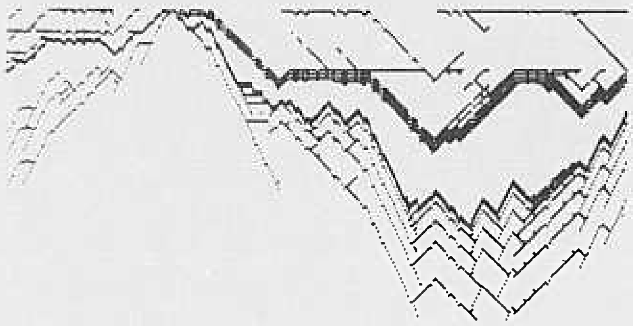
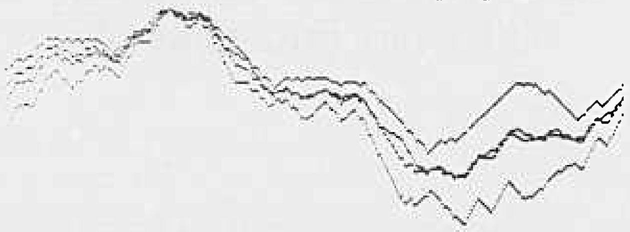
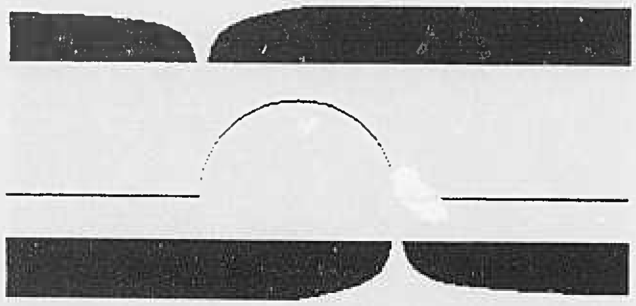


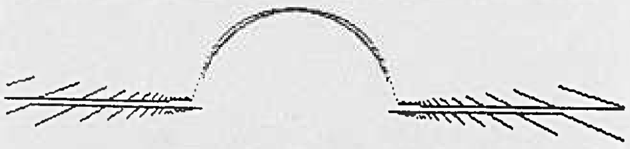
Figure 4

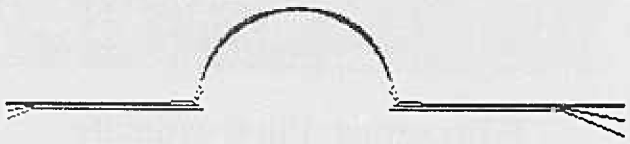

Second series: Original curve and suntrace

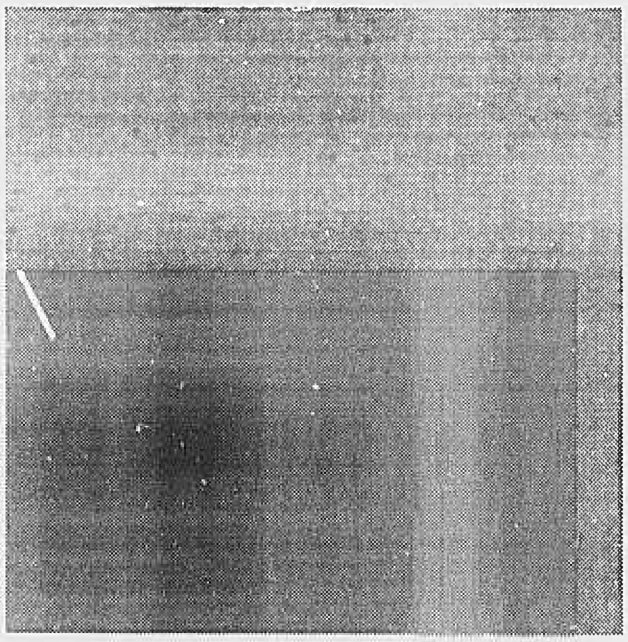

Second series: Constraint propagation

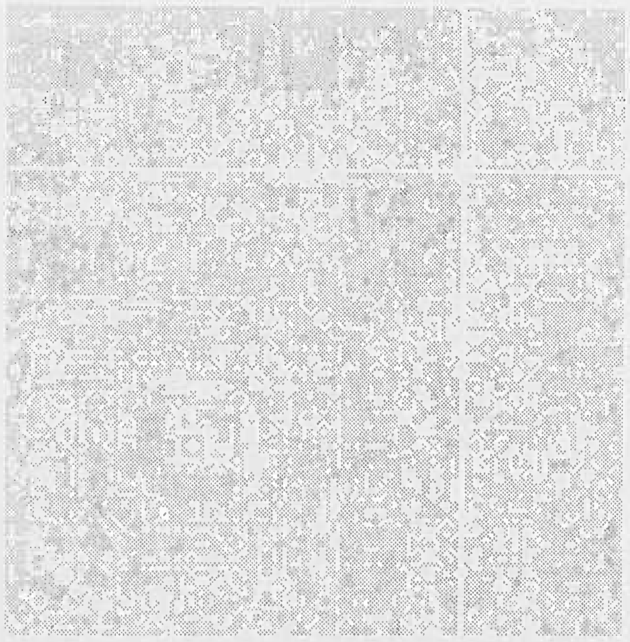

Second series: Final bounds and estimate

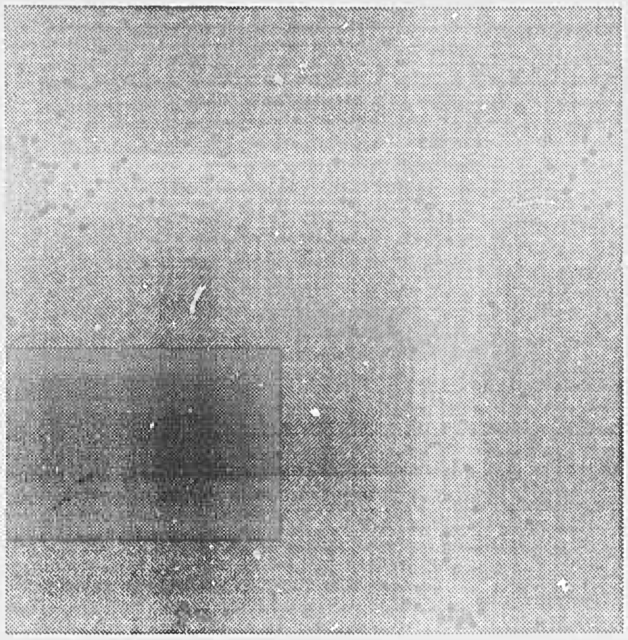

Third series: Original curve and suntrace

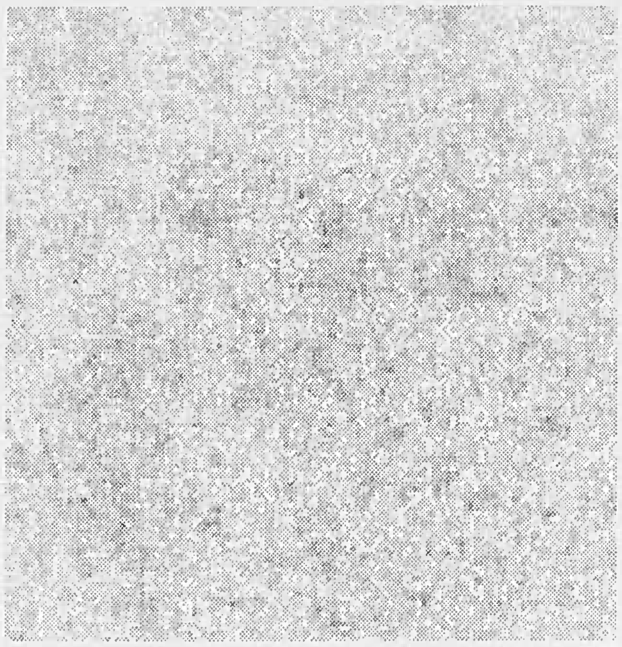

Third series: Constraint propagation

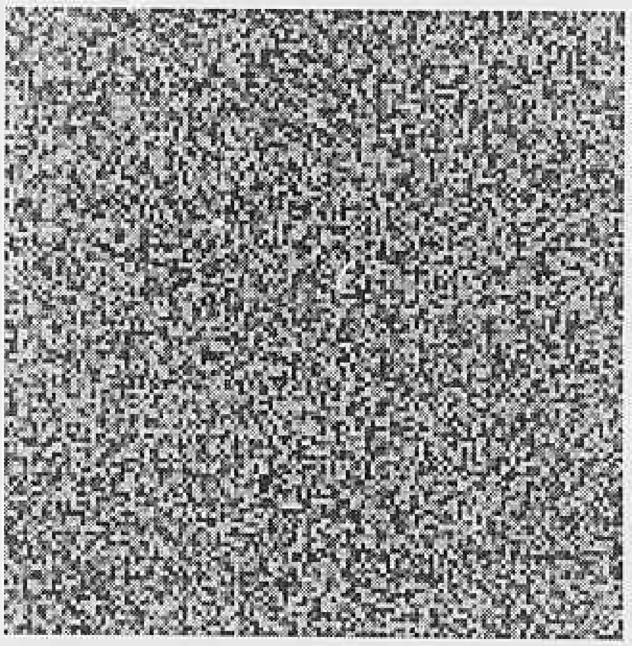

Third series: Final bounds and estimate

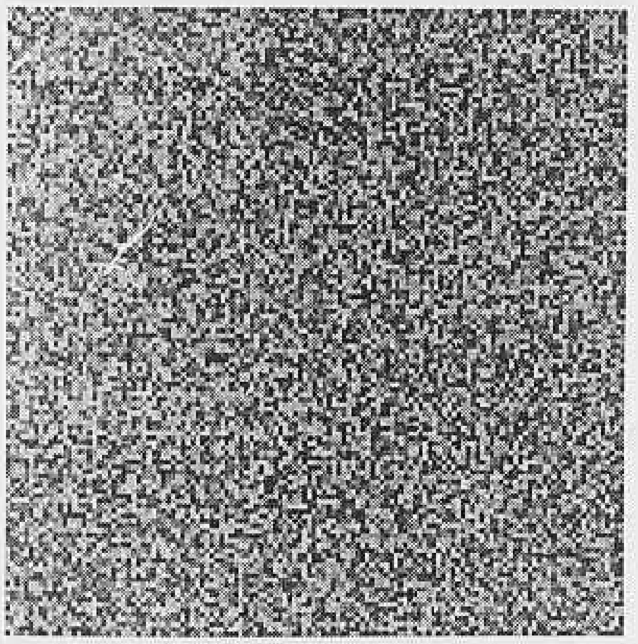

Fourth series: Original surface


Fourth series: Error times 10


Fourth series: Final estimate


Fifth series: Error times 10


Fifth series: Original surface


Fifth series: Final estimate